AUTARQUIA ASSOCIADA À UNIVERSIDADE DE SÃO PAULO

Modelagem de um sistema de planejamento em Radioterapia e Medicina Nuclear com o uso do código MCNP6

Felipe Massicano

Tese apresentada como parte dos requisitos para obtenção do Grau de Doutor em Ciências na Área de Tecnologia Nuclear - Reatores

Orientador:
Prof. Dr. Hélio Yoriyaz

Coorientador:
Prof. Dr. Marcel Parolin Jackowski

São Paulo
2015

INSTITUTO DE PESQUISAS ENERGÉTICAS E NUCLEARES
Autarquia associada à Universidade de São Paulo

Modelagem de um sistema de planejamento em Radioterapia e Medicina Nuclear com o uso do código MCNP6

Felipe Massicano

Tese apresentada como parte dos requisitos para obtenção do Grau de Doutor em Ciências na Área de Tecnologia Nuclear - Reatores

Orientador:
Prof. Dr. Hélio Yoriyaz

Coorientador:
Prof. Dr. Marcel Parolin Jackowski

Versão Corrigida
Versão Original disponível no IPEN

São Paulo
2015

Ao meu pai, ***Edson Antônio Massicano****, e à minha mãe,* ***Hilda Lídia Farina Massicano****, por sempre terem apoiado os meus sonhos. Às minhas irmãs* ***Renata Massicano Borlenghi*** *e* ***Daniela Massicano Fichera*** *que são parte do que sou hoje. E à minha esposa,* ***Adriana Vidal Fernandes Massicano****, pelo amor, dedicação, carinho, confiânça e ter me incentivado a jamais desistir.*

Agradecimentos

À **Deus** pois sem Ele absolutamente nada foi, é ou será possível.

Aos meus pais, pelo carinho, exemplo, amor, fé, guia... "Nada se pode ensinar a um homem; você pode apenas ajudá-lo a encontrar a resposta dentro dele mesmo" (*Galileu Galilei*).

Às minhas irmãs Renata e Daniela, pelos risos, pelas lembranças, pelos sobrinhos, pela infância... pela vida. "O valor das coisas não está no tempo que elas duram, mas na intensidade com que acontecem. Por isso existem momentos inesquecíveis, coisas inexplicáveis e pessoas incomparáveis" (*Fernando Sabino*).

À minha eterna esposa, Adriana Vidal Fernandes Massicano, por tudo o que temos vivido nestes 7 anos de casamento e por tudo que ainda viveremos. "Depois de um tempo você aprende que o que importa não é o que você tem na vida, mas quem você tem na vida" (*William Shakespeare*).

Ao meu orientador Dr. Hélio Yoriyaz, pela oportunidade, pelo conhecimento transmitido, pelo auxílio sempre presente e pela amizade. Por ter me permitido trabalhar e de nunca ter recusado ajuda. "Mestre não é aquele que aprendeu a ensinar, mas aquele que ensina a aprender" (*Marcelo Soriano*).

Ao meu coorientador Dr. Marcel Parolin Jackowski por sua ajuda, conselho e incentivo imprescindíveis à minha formação.

Ao grupo de Informática Médica do A.C.Camargo Cancer Center, onde tive, por dois anos, não somente a oportunidade de trabalhar mas também de aprender muito. Sou grato a esse grupo e a todos vinculados a ele pela paciência em me ensinar e pela amizade conseguida. Em especial queria agradecer ao Dr. Diogo da Costa Patrão, por ter apostado em mim e me inserido no grupo de Informática Médica. Também agradeço ao

Marcelo Sagayama, Michel Oleynik, Ricardo Moura Sekeff Budaruiche, Calebre Rodrigues de Nobrega, Aline Damascena e Fabio Rampazzo Mathias por terem sido meus parceiros, sempre atenciosos e principalmente amigos.

Aos amigos do Centro de Engenharia Nuclear por todos os momentos descontraídos, pelo profissionalismo, pela ajuda... "Um Amigo se faz rapidamente; já a amizade é um fruto que amadurece lentamente." (*Aristóteles*).

Ao Instituto de Pesquisas Energéticas e Nucleares, em especial o Centro de Engenharia Nuclear pelo provimento dos recursos necessário para o desenvolvimento e divulgação deste trabalho. Aos funcionários da divisão de pós-graduação do IPEN.

Aos demais amigos e familiares de todas as partes.

"*Não se aparte da tua boca o livro desta lei; antes medita nele dia e noite, para que tenhas cuidado de fazer conforme a tudo quanto nele está escrito; porque então farás prosperar o teu caminho, e então prudentemente te conduzirás*"

Josué 1:8

Resumo

O tratamento de câncer possui diversas modalidades. Uma delas é a utilização de fontes de radiação como principal protagonista do tratamento. A radioterapia e a medicina nuclear são exemplos desse tipo de tratamento. Por utilizarem a radiação ionizante como principal ferramenta para a terapia, há a necessidade de se efetuar diversas simulações do tratamento a fim de maximizar a dose nos tecidos tumorais sem ultrapassar os limites de dose nos tecidos sadios circunvizinhos. Os sistemas utilizados na simulação desses tipos de terapia recebem o nome de Sistemas de Planejamento Dosimétrico. A medicina nuclear e a radioterapia possuem seus próprios sistemas de planejamento dosimétricos devido a grande diversidade das informações necessárias às suas simulações. Os sistemas de planejamento em radioterapia são mais consolidados do que os de medicina nuclear e por tal motivo um sistema que aborde tanto os casos de radioterapia como de medicina nuclear contribuiria para significativos avanços na área de medicina nuclear. Dessa forma, o objetivo do trabalho foi modelar um Sistema de Planejamento Dosimétrico com o uso do código de Monte Carlo MCNP6 *Monte Carlo N-Particle Transport Code* que permitisse incorporar os casos de radioterapia e medicina nuclear e que fosse extensível a novos tipos de tratamentos. A modelagem desse sistema resultou na construção de um *Framework*, orientado a objetos, nomeado IBMC o qual auxilia no desenvolvimento de sistemas de planejamento que necessitam interpretar grandes quantidades de informações com o objetivo de escrever o arquivo base do *MCNP6*. O IBMC permitiu desenvolver de maneira rápida e prática sistemas de planejamento para radioterapia e medicina nuclear e os resultados foram validados com sistemas já consolidados. Ele também mostrou alto potencial para desenvolver sistemas de planejamento de novos tipos de tratamentos que utilizam a radiação ionizante.

Abstract

Cancer therapy has many branches and one of them is the use of radiation sources as treatment leading method. Radiotherapy and nuclear medicine are examples of these treatment types. For using the ionization radiation as main tool for the therapy, there is the need of crafting many treatment simulation in order to maximum the tumoral tissue dose without throught the dose limit in health tissue surrounding. Treatment planning systems (TPS) are systems which have the purpose of simulating these therapy types. Nuclear medicine and radiotherapy have many distinct features linked to the therapy mode and consequently they have different TPS destined for each. The radiotherapy TPS is more developed than the nuclear medicine TPS and by that reason the development of a TPS that was similar to the radiotherapy TPS, but enough generic for include other therapy types, it will contribute with significant advances in nuclear medicine and in others therapy types with radiation. Based on this, the goal of work was to model a TPS that utilizes the Monte Carlo N-Particle Transport code (MCNP6) in order to simulate radiotherapy therapy, nuclear medicine therapy and with potential for simulating other therapy types too. The result of this work was the creation of a *Framework* in Java language, object-oriented, named IBMC which will assist in the development of new TPS with MCNP6 code. The IBMC allowed to develop rapidly and easily TPS for radiotherapy and nuclear medicine and the results were validated with systems already consolidated. The IBMC showed high potential for developing TPS by new therapy types.

Sumário

1. Introdução 19

2. Objetivo 21

3. Revisão Bibliográfica 22

3.1 Breve histórico do uso da radiação ionizante em tratamentos de câncer. . . 22

3.2 Radioterapia 22

3.2.1 Teleterapia 23

3.2.2 Braquiterapia 23

3.3 Medicina Nuclear 24

3.4 Planejamento do Tratamento 25

3.4.1 Sistemas de planejamento do tratamento com códigos de Monte Carlo em Medicina Nuclear 27

3.4.1.1 Softwares utilizados em Sistema de Planejamento com códigos de Monte Carlo 28

3.4.2 Sistemas de planejamento do tratamento com códigos de Monte Carlo em Radioterapia 29

3.4.2.1 Caracterização da fonte de radiação 29

3.4.2.2 Fatores geométricos do tratamento 31

3.4.2.3 Informações pertinentes aos códigos de Monte Carlo. . . . 32

3.4.2.4 Softwares utilizados em Sistema de Planejamento com códigos de Monte Carlo 33

3.5 Estrutura de sistemas modulares de planejamento baseados em códigos de Monte Carlo. 34

4. Materiais e Métodos 37
4.1 Materiais 37
4.1.1 MCNP6 - *Monte Carlo N-Particle Transport Code* 37
4.1.2 AMIGO *(Algorithm for Medical Image Generating Object)* 37
4.1.3 AMIGOBrachy *A Medical Image-based Graphical platfOrmd - Brachytherapy module* 38
4.1.4 ICCT - *Image Converter Computed Tomography* 38
4.1.5 Objeto Simulador Antropomórfico *RANDO® Phantoms* 38
4.1.6 Java 39
4.1.7 Conceitos de Orientação a Objetos 39
4.1.7.1 Classes e Objetos 40
4.1.7.2 Herança 40
4.1.7.3 Polimorfismo 40
4.1.8 Padrões de Projetos 40
4.1.8.1 Padrão - *Visitor* 41
4.2 Métodos 42
4.2.1 Modelo genérico de um Sistema de Planejamento Dosimétrico que utiliza o código *MCNP* 42
4.2.2 IBMC (*Input Builder of Monte Carlo Code*): Constrói arquivos de entrada para códigos de Monte Carlo 43
4.2.3 Módulo de Construção. 45
4.2.4 Módulo de Dados 49
4.2.5 Vínculo entre os Módulos de Dados e de Construção 50
4.2.6 Exemplo da execução do IBMC para criação do arquivo de entrada do *MCNP*. 50

5. Resultados e Discussões 53
5.1 Exemplos de utilização do IBMC 53
5.1.1 Medicina Nuclear 53
5.1.2 Braquiterapia 73
5.1.3 Teleterapia 84

5.1.4 Exemplo de uma possível integração de IBMC com o *ICCT* a fim de gerar a geometria de um paciente 96
5.2 Casos de uso com o IBMC 100
5.2.1 Primeiro caso de uso: Geração de arquivos complexos do *MCNP* por usuários iniciantes 100
5.2.2 Segundo caso de uso: Adaptar um sistema baseado no IBMC a medida que surgem novas tecnologias 102
5.2.3 Terceiro caso de uso: Usar um sistema baseado no IBMC para comparar diferentes métodos 102

6. Conclusões 104

Apêndice 106

A. Estrutura do arquivo de configuração do problema (INP) do MCNP 107

B. Diagramas de Classes 108

C. Biblioteca helperMCNP presente no IBMC. 110
C.1 Principais características da biblioteca *helperMCNP.* 110
C.1.1 Construção de Superfícies. 110
C.1.2 Construção de Células. 119
C.1.3 Fonte de Radiação. 143
C.1.4 Construção dos *Tally.* 166
C.1.5 Definição dos Materiais. 177
C.1.6 Características da física da simulação. 182
C.1.7 Arquivo de saída do *MCNP.* 191
C.1.8 Depurar simulação. 196
C.1.9 Finalização da simulação. 198

Referências 200

Lista de Tabelas

3.1 Softwares baseados em métodos de Monte Carlo para cálculo de dose em Medicina Nuclear 28
3.2 Softwares baseados em códigos de Monte Carlo para Radioterapia 33

4.1 Descrição dos cinco métodos concretos e quatro abstratos da classe *Abstract-InputMonteCarloCode*. 46
4.2 Descrição dos principais métodos da classe *InputMCNP* presente na Figura 4.4. 48
4.3 Representação das descrições das classes referentes aos Módulo de Dados. 49
4.4 Descrição dos métodos referente à classe *SimulationFileManager*. 51

5.1 Composição dos materiais referente à simulação de Medicina Nuclear . . . 55
5.2 Descrição dos métodos da classe *GeometryDataMCNP*. 57
5.3 Descrição dos métodos da classe *SourceDataMCNP*. 59
5.4 Descrição dos métodos da classe *ConfigDataMCNP*. 62
5.5 Descrição dos métodos da classe *RepeatedStructureForPhantom*. 66
5.6 Descrição dos métodos da classe *RepeatedStructureForSource*. 69
5.7 Composição dos materiais referente à simulação de Braquiterapia. 75
5.8 Composição dos materiais referente à simulação de Teleterapia. 86

C.1 Descrição dos principais métodos da classe *Surface*. 111
C.2 Parâmetros para a construção de um cilindro no *MCNP*. 111
C.3 Parâmetros para a construção de um plano no *MCNP*. 113
C.4 Parâmetros para a construção de uma esfera no *MCNP*. 114
C.5 Parâmetros para a construção de um cone no *MCNP*. 115

C.6 Parâmetros para a construção de uma elipsoide hiperbólica paraboloide no *MCNP*. 117
C.7 Descrição dos parâmetros necessários para se criar uma célula conforme a primeira especificação. 120
C.8 Descrição dos parâmetros do cartão *SI*. 144
C.9 Descrição dos parâmetros do cartão SP. 150
C.10 Descrição dos parâmetros do cartão *SDEF* que estão contidos em *helperMCNP*. 158
C.11 Descrição dos parâmetros do cartão *SSR* que estão contidos em *helperMCNP*. 162
C.12 Descrição dos parâmetros dos cartões *DE* e *DF* que estão contidos em *helperMCNP*. 167
C.13 Descrição dos parâmetros de *FM* que estão contidos em *helperMCNP*. . . . 169
C.14 Descrição dos parâmetros do *Tally FMESH* que estão contidos em *helperMCNP*. 171
C.15 Descrição dos parâmetros do *Tally F8* que estão contidos em *helperMCNP*. 175
C.16 Descrição dos parâmetros do Cartão *M*. 177
C.17 Descrição dos parâmetros do cartão *PHYS* para fótons. 184
C.18 Descrição dos parâmetros do cartão *PHYS* para elétrons. 187
C.19 Descrição dos parâmetros do Cartão *CUT*. 189
C.20 Descrição dos parâmetros do Cartão *PRDMP*. 193

Lista de Figuras

3.1 Número de artigos que possuem o termo "*Monte Carlo*" em publicações com os termos "*Radiotherapy*" e "*Nuclear Medicine*" ou "*Radiopharmaceuticals*". 26

3.2 Número de artigos que possuem o termo "*Radiopharmaceuticals*" em publicações com os termos "*Therapy*" e "*Dosimetry*". 26

3.3 Esquema do cabeçote de um acelerador linear Elekta SLi plus. 30

4.1 Modelo genérico dos Sistemas de Planejamentos com o uso do código de Monte Carlo MCNP. 42

4.2 Modelo do fluxo dos dados estabelecido pelo IBMC. 44

4.3 Diagrama de classes correspondente aos módulos de construção. 46

4.4 Diagrama de Classes para os módulos de construção referente ao *MCNP*. . 48

4.5 Diagrama de Classes que representa o vínculo entre os Módulos de Dados e Construção. 50

4.6 Diagrama da classe de *SimulationFileManager*. 51

4.7 Diagrama de sequência para a criação de um arquivo *INP* do *MCNP* através do IBMC. 52

5.1 Esquema da geometria do problema de medicina nuclear. 55

5.2 Diagrama de classes de *GeometryDataMCNP*. 56

5.3 Diagrama de sequência que exemplifica o uso da classe *GeometryDataMCNP* para extrair as informações provenientes do *software* terceiro. 58

5.4 Diagrama de classes de *SourceDataMCNP*. 59

5.5 Diagrama de sequência que exemplifica o uso da classe *SourceDataMCNP* para extrair as informações do arquivo *source*. 61

5.6 Diagrama de classes de *ConfigDataMCNP*. 62
5.7 Esquema da formato do arquivo de configuração que contêm informações para os cartões *NPS* , *PRDMP* e *PRINT*. 63
5.8 Diagrama de sequência que exemplifica o uso da classe *ConfigDataMCNP* para extrair as informações do arquivo *tally* e *config*. 64
5.9 Diagrama de sequência da implementação do método *message*. 64
5.10 Diagrama de sequência da implementação do método *header*. 65
5.11 Diagrama de classe de *RepeatedStructureForPhantom*. 65
5.12 Diagrama de sequência da implementação do método *cellBlock*. 68
5.13 Diagrama de sequência da implementação do método *surfaceBlock*. 68
5.14 Diagrama de classe de *RepeatedStructureForSource*. 69
5.15 Diagrama de sequência da implementação do método *dataBlock*. 71
5.16 Diagrama de sequência da implementação do método *commentBlock*. . . . 71
5.17 Diagrama de sequência que representa os passos para gerar o arquivo *INP* do *MCNP*, com o IBMC, para o caso de Medicina Nuclear. 72
5.18 Representação da geometria de Braquiterapia. 74
5.19 Fluxo de dados para a criação dos arquivos do *MCNP* com o uso do *software AMIGOBrachy* e o IBMC. 75
5.20 Diagrama de classes de *GeometryDataMCNP* para o caso de Braquiterapia. 76
5.21 Diagrama de sequência do preenchimento de um objeto da classe *GeometryDataMCNP* para o caso de Braquiterapia. 77
5.22 Diagrama de classes de *GeometryDataMCNP* para o caso de Braquiterapia. 78
5.23 Diagrama de sequência do preenchimento de um objeto da classe *SourceDataMCNP* para o caso de Braquiterapia. 79
5.24 Diagrama de classes de *ConfigDataMCNP* para o caso de Braquiterapia. . 80
5.25 Diagrama de sequência do preenchimento de um objeto da classe *ConfigDataMCNP* para o caso de Braquiterapia. 81
5.26 Diagrama de sequência referente à criação do arquivo de entrada para o *MCNP* para o caso de Braquiterapia. 83
5.27 Imagem da geometria do paciente, no caso de Braquiterapia, criado com o auxílio do IBMC. Em (a) tem-se um corte transversal do paciente e em (b) um corte coronal. 84

5.28 Representação da geometria de Teleterapia. 86
5.29 Diagrama de sequência do preenchimento de um objeto da classe *GeometryDataMCNP* para o caso de Teleterapia. 88
5.30 Diagrama de sequência do preenchimento de um objeto da classe *SourceDataMCNP* para o caso de Teleterapia. 90
5.31 Diagrama de sequência do preenchimento de um objeto da classe *ConfigDataMCNP* para o caso de Teleterapia. 92
5.32 Diagrama de sequência referente à construção do arquivo do MCNP para o caso de Teleterapia. 94
5.33 Geometria do paciente, criada com o auxílio do IBMC, para o caso de Teleterapia. Em (a) é apresentado um corte transversal do paciente e em (b) um corte transversal. 95
5.34 Diagrama de classes de *GeometryDataMCNP* somente com os métodos utilizados para o caso da integração do IBMC com o ICCT. 97
5.35 Diagrama de sequência referente preenchimento do objeto ***geo*** da classe *GeometryDataMCNP* mediante os dados do *ICCT*. 98
5.36 Diagrama de sequência referente à criação do arquivo de entrada do MCNP por intermédio da classe *Icct*. 99
5.37 Imagem do objeto simulador RANDO® criado no MCNP por intermédio da classe *Icct*. 99
5.38 Esquema do IBMC feito pelo usuário experiente. 101

A.1 Estrutura do arquivo de configuração da simulação no MCNP. 107

B.1 Representação de uma classe concreta no diagrama de classes. 108
B.2 Representação de uma classe abstrata no diagrama de classes. 108
B.3 Representação de uma interface no diagrama de classes. 109
B.4 Representação de relações de herança e implementação no diagrama de Classes. 109
B.5 Representação de relações de agregação e composição no diagrama de Classes. 109

C.1 Diagrama de classes de *Surface*. 111
C.2 Diagrama da classe *Cx*. 112

C.3 Diagrama de sequência para exemplificar desde a criação de um objeto da classe *Cx* até o retorno de sua especificação no *MCNP*. 112
C.4 Diagrama das classes *Px*, *Py* e *Pz*. 113
C.5 Diagrama de sequência que exemplifica a criação de objetos de *Px*, *Py* e *Pz* e a extração de suas respectivas saídas para o *MCNP*. 114
C.6 Diagrama de classes de *So*. 115
C.7 Diagrama de sequência caracterizando a criação do objeto de *So* e a requisição de sua representação para o *MCNP*. 115
C.8 Diagrama de classes de *Kx*. 116
C.9 Diagrama de sequência caracterizando a criação do objeto de *Kx* e a requisição de sua representação para o *MCNP*. 117
C.10 Diagrama de classes de *Sq*. 118
C.11 Diagrama de sequência caracterizando a criação do objeto de *Sq* e a requisição de sua representação para o *MCNP*. 119
C.12 Diagrama de classes de *CellSpecification*. 121
C.13 Diagrama de classes referente à classe *GroupSurface*. 122
C.14 Diagrama de sequência que exemplifica o uso da classe *Intersection*. 123
C.15 Diagrama de sequência que exemplifica o uso da classe *Union*. 124
C.16 Diagrama de sequência que exemplifica o uso da classe *ComplexGroupSurface*. 125
C.17 Diagrama de classes relacionado à *Complement*. 126
C.18 Diagrama de sequência relacionado à *CellComplement*. 127
C.19 Diagrama de sequência relacionado à *SurfaceComplement*. 127
C.20 Diagrama de classes concernente aos possíveis parâmetros de uma célula. . 128
C.21 Diagrama de sequência da criação de *Universe*. 129
C.22 Diagrama de sequência da criação do *Lattice*. 130
C.23 Diagrama de sequência da criação de *Fill*. 130
C.24 Diagrama de sequência da criação de *FillFull*. 131
C.25 A imagem apresenta um cubo formado por *voxels*. 132
C.26 Diagrama de sequência da criação de *Importance*. 133
C.27 Diagrama de sequência da criação de uma célula conforme o primeiro exemplo. 135
C.28 Diagrama de sequência da criação de um célula conforme o segundo exemplo. 137

C.29 Diagrama de sequência da criação de um célula conforme o terceiro exemplo. 138
C.30 Diagrama de sequência da criação de um célula conforme o quarto exemplo. 140
C.31 Diagrama de sequência da criação de um célula conforme o quinto exemplo. 141
C.32 Diagrama de classes referente à *CellsLikeBut*. 142
C.33 Diagrama de sequência referente ao uso da classe *CellsLikeBut*. 143
C.34 Diagrama de classes de *SI*, *SP* e *SI_SP*. 146
C.35 Formas possíveis de representar o cartão *SI*. 146
C.36 Diagrama de classes de *SI* e sua relação com *SI_Values* o qual representa os valores de *SI*. 147
C.37 Diagrama de sequência que exemplifica a criação do cartão *SI* com valores numéricos pela biblioteca *helperMCNP*. 148
C.38 Diagrama de sequência que exemplifica a criação do cartão *SI* com valores correspondentes às Estruturas Repetidas. 150
C.39 Relação entre o cartão *SP* e *SI*. 151
C.40 Diagrama de classes de *SP* e sua relação com *SP_Values* o qual representa os valores de *SP*. 152
C.41 Diagrama de sequência que exemplifica a criação do cartão *SP* com seus valores correlacionados com os valores numéricos de *SI*. 153
C.42 Diagrama de sequência que exemplifica a criação do cartão *SP* com seus valores correlacionados aos valores de *SI* especificados com relação à Estruturas Repetidas. 154
C.43 Diagrama de classes de *SI_SP*. 155
C.44 Diagrama de sequência que exemplifica o uso da classe *SI_SP* onde tanto os valores de *SI* como *SP* são números. 156
C.45 Diagrama de sequência que exemplifica o uso da classe *SI_SP* onde os valores de *SI* são representados com base em Estruturas Repetidas. 158
C.46 Diagrama de classes referente ao cartão *SDEF*. 159
C.47 Diagrama de sequência que corresponde a criação de uma fonte de fótons. . 161
C.48 Diagrama de classes referente à criação do cartão *SSR*. 162
C.49 Diagrama de sequencia referente à criação do cartão *SSR*. 164
C.50 Diagrama de classes referente ao cartão *Mode*. 165
C.51 Diagrama de sequência para exemplificar a criação do cartão *Mode*. 166

C.52 Diagrama de classes de *DE_DF*. 168
C.53 Diagrama de sequência para exemplificar o uso da classe *DE_DF*. 169
C.54 Diagrama de classes de *FM*. 170
C.55 Diagrama de sequência para exemplificar o uso da classe *FM*. 171
C.56 Diagrama de classes de *FMESH*. 173
C.57 Diagrama de sequência para exemplificar o uso da classe *FMESH*. 174
C.58 Diagrama de classes correspondentes à classe *F8*. 176
C.59 Diagrama de sequência para exemplificar a criação do *Tally F8* pela biblioteca *helperMCNP*. 176
C.60 Diagrama de Classes de *MaterialMCNP* e *GroupOfMaterialMCNP*. 180
C.61 Diagrama de sequência que exemplifica o uso da classe *MaterialMCNP*. . . 181
C.62 Diagrama de sequência que exemplifica o uso da classe *GroupOfMaterialMCNP*. 182
C.63 Diagrama de classe de *PHYS* com suas subclasses *PhysPhoton* e *PhysElectron*.184
C.64 Diagrama de sequência para criar o cartão *PHYS* para fótons através da classe *PhysPhoton*. 186
C.65 Diagrama de sequência para criar o cartão *PHYS* para elétrons mediante a classe *PhysElectron*. 189
C.66 Diagrama de classe correspondente ao cartão *CUT*. 190
C.67 Diagrama de sequência para exemplificar o uso da classe *CUT*. 191
C.68 Diagrama de classe correspondente ao cartão *PRINT*. 192
C.69 Diagrama de sequência para exemplificar o uso da classe *PRINT*. 193
C.70 Diagrama de classe correspondente ao cartão *PRDMP*. 195
C.71 Diagrama de sequência para exemplificar o uso da classe *PRDMP*. 196
C.72 Diagrama de classe correspondente ao cartão *DBCN*. 197
C.73 Diagrama de sequência para exemplificar o uso da classe *DBCN*. 198
C.74 Diagrama de classe correspondente ao cartão *NPS*. 199
C.75 Diagrama de sequência para exemplificar o uso da classe *NPS*. 199

Capítulo 1

Introdução

O tratamento de câncer possui diversas modalidades. Uma delas é a utilização de fontes de radiação como principal protagonista do tratamento. A Radioterapia e a Medicina Nuclear são exemplos desse tipo de tratamento. Por utilizarem a radiação ionizante como principal ferramenta para a terapia, há a necessidade de se efetuar diversas simulações do tratamento antes de ser efetivamente realizado. Essas simulações tem por finalidade ajudar o físico médico a encontrar os melhores parâmetros, específicos a cada terapia, e assim maximizar a dose nos tecidos tumorais sem prejudicar o tecido sadio (1). Os sistemas utilizados para simularem esse tipo de terapia recebem o nome de Sistemas de Planejamento Dosimétrico.

Sendo assim, os sistemas de planejamento são *softwares* que recebem diversas informações específicas de um determinado tratamento e fornecem a distribuição de dose no corpo do paciente. Para obter uma estimativa da dose de maneira fidedigna, esses sistemas utilizam modelos físicos de transporte de radiação, sendo que, tais modelos necessitam da correta caracterização da fonte de radiação e também de fatores geométricos envolvidos no tratamento, como por exemplo a geometria do paciente.

Geralmente cada tipo de tratamento possui seu próprio sistema de planejamento, isto por causa de grandes diversidades encontradas entre os procedimentos terapêuticos. A caracterização da fonte é a principal responsável por essas diversidades. Por exemplo, em teleterapia, uma modalidade da radioterapia, a caracterização da fonte de radiação é geralmente realizada através da simulação do cabeçote do acelerador onde é gerado o feixe de radiação seja elétrons ou fótons. Na braquiterapia, outra modalidade da radioterapia, ela está relacionada à definição de fontes seladas, já em medicina nuclear ela está interna ao paciente (2).

Outro fator que atua como grande responsável pelas diversidades entre os sistemas de planejamento são os modelos físicos de transporte de radiação. A Radioterapia utiliza principalmente o modelo do "*Pencil Beam Algorithm*"(3), a Braquiterapia se baseia no protocolo TG-43U1 - (*Task Group-No 43 Report-Updated*)(4), a Medicina Nuclear possui o sistema MIRD -(*Committee on Medical Internal Radiation Dose*)(5).

Porém uma informação que poderia ser compartilhada entre os diversos sistemas seria a caracterização da geometria do paciente, em que alguns sistemas de planejamento utilizam imagens tomográficas tais como tomografia computadorizada (CT) ou ressonância magnética (MRI), para adquirir essas informações (6, 7, 8, 9).

Dessa forma, um sistema que seja capaz de abranger todas essas modalidades de tratamento deve ser robusto, na estimativa da dose, e flexível, para lidar com as diversidades entre os diferentes tipos de tratamentos (10).

Capítulo 2

Objetivo

O objetivo desse trabalho é modelar um Sistema de Planejamento Dosimétrico utilizando o código de Monte Carlo MCNP6 *Monte Carlo N-Particle Transport Code*, que permita incorporar as técnicas de radioterapia e medicina nuclear e que seja extensível a novos tipos de procedimentos terapêuticos.

Um modelo de sistema de planejamento que permita a construção automática da distribuição da fonte de radiação mediante imagens de medicina nuclear é um fator inédito no presente trabalho. Por intermédio de tais objetivos, este sistema permitirá o estudo e o acompanhamento dosimétrico de diversos casos clínicos em medicina nuclear e radioterapia.

Capítulo 3

Revisão Bibliográfica

Neste capítulo apresentam-se as revisões bibliográficas referente à Medicina Nuclear e Radioterapia bem como os planejamentos dosimétricos que utilizam código de Monte Carlo.

3.1 Breve histórico do uso da radiação ionizante em tratamentos de câncer.

A física radiológica é a ciência que estuda a radiação ionizante e sua interação com a matéria tendo com o principal interesse a energia absorvida pela matéria (11).

O início da física radiológica se deu com a descoberta dos raios-X pelo físico alemão Wilhelm Conrad Röntgen. Em seguida houve a descoberta da radioatividade por Henri Becquerel e do rádio por Marie Curie e Pierre Curie (11).

Tais descobertas, além de revolucionar o entendimento da física na comunidade científica, também contribuiram grandemente para o desenvolvimento da medicina, com o uso por exemplo, de raios-X no tratamento de câncer já em 1896 (11).

Desde então, houve o surgimento de diversas tecnologias que contribuíram para o avanço das duas grandes modalidades de tratamento contra o câncer: a medicina nuclear e a radioterapia.

3.2 Radioterapia

A radioterapia é uma modalidade de tratamento que envolve o uso de radiações ionizantes. O seu uso mais comum é em tratamentos de câncer(12, 13, 14).

A radioterapia possui duas modalidades a teleterapia e a braquiterapia descritas a seguir.

3.2.1 Teleterapia

A teleterapia realiza os tratamentos mediante equipamentos que direcionam feixes de radiações ionizantes na área tumoral. Esses feixes de radiação podem ser de diversos tipos, como por exemplo, de fótons ou elétrons (12, 14, 15).

O surgimento da teleterapia iniciou-se no ano de 1945 com a produção de equipamentos de megavoltagem e unidades de teleterapia. A primeira instalação de uma unidade de cobalto 60 data de 1951 sendo seu uso difundido por diversos centros. Nessas unidades a fonte de radiação são isótopos que emitem raios gama (12, 14).

Basicamente há dois tipos de equipamentos de teleterapia. Os teleisótopos que emitem raios gama através de fontes de cobalto 60 e os aceleradores lineares que podem emitir diversos tipos feixes de radiação, como por exemplo, de fótons, elétrons ou até mesmo prótons por intermédio da aceleração de partículas carregadas (12, 14).

Devido a grandes avanços com relação aos aceleradores lineares, eles se tornaram os mais utilizados em radioterapia (15).

3.2.2 Braquiterapia

A Braquiterapia é uma modalidade da radioterapia, que utiliza pequenas fontes de radiação encapsuladas que são colocadas a uma curta distância ou mesmo em contato com a lesão a ser tratada. A dose é então entregue continuamente, tanto sobre um curto período de tempo ou até que a fonte de radiação decaia totalmente (15, 16).

Há duas principais de técnicas em braquiterapia. A primeira é o tratamento intracavitário em que as fonte são colocadas dentro de cavidades do corpo e próximas ao volume tumoral. O segunda é o intersticial em que as fontes são implantadas dentro da região tumoral (15).

Os tratamentos intracavitários são sempre temporários e de curta duração, já os tratamentos intersticiais podem tanto serem temporários como permanentes (15).

A vantagem da braquiterapia em relação à teleterapia é a alta precisão na irradiação do volume tumoral. No entanto ela possui a desvantagem de poder somente ser utilizada em casos nos quais o tumor é bem localizado e relativamente pequeno (15).

3.3 Medicina Nuclear

A Medicina Nuclear é uma modalidade médica que se utiliza de radiofármacos para fins diagnósticos ou terapêuticos.

Os exames em Medicina Nuclear são realizados através da administração do radiofármaco ao paciente por diferentes vias sendo a endovenosa a mais utilizada (17, 18).

Segundo a Resolução RDC nº63 (19), os radiofármacos: "são preparações farmacêuticas com finalidade diagnóstica ou terapêutica que, quando prontas para o uso, contêm um ou mais radionuclídeos. Compreendem também os componentes não-radioativos para marcação e os radionuclídeos, incluindo os componentes extraídos dos geradores de radionuclídeos".

Conforme mesma Resolução RDC nº63 (19), os isótopos radioativos ou radionuclídeos: "são isótopos instáveis os quais sofrem decaimento radioativo e transmutam-se em novo elemento. São átomos que se desintegram por emissão de radiação corpuscular (partícula) ou eletromagnética".

Um exemplo muito difundido na Medicina Nuclear, que tem como finalidade o diagnóstico, é a administração no paciente, na ordem de micro grama, de FDG (2-Flúor 2-Desoxi D-Glicose) marcado com o elemento radioativo ^{18}F, emissor de pósitron (20, 21, 22, 23). Tal procedimento permite o diagnóstico precoce de tumores (24, 25, 26).

Em casos de terapia ainda é muito comum o uso de ^{131}I que tem como principal objetivo o tratamento de doenças tireoidianas (27, 28, 29). Ao ser incorporado no organismo o ^{131}I sofre grande captação pelas células da tireóide onde efetuará o tratamento. O tratamento é realizado pela irradiação do tecido tumoral mediante a emissão de partículas *beta* provenientes do decaimento do elemento ^{131}I (30).

Também, há o crescimento em Medicina Nuclear de uma nova modalidade de diagnóstico e terapia as quais utilizam anticorpos monoclonais (AcM), radioimunodiagnóstico e radioimunoterapia respectivamente (31, 32, 33). Os anticorpos monoclonais acoplados aos radioisótopos combinam os efeitos biológicos dos anticorpos com os efeitos radiolíticos dos radioisótopos a fim de destruir as células tumorais (34, 35, 36, 37).

3.4 Planejamento do Tratamento

O principal objetivo do planejamento do tratamento é estudar a melhor maneira de se maximizar a dose nos tecidos tumorais sem ultrapassar os limites de dose nos tecidos sadios, principalmente os órgãos críticos (12, 38).

O planejamento pode ser realizado manualmente ou através de *softwares*. Nas áreas de medicina nuclear e radioterapia há diversos *softwares* que simulam a terapia com o objetivo não somente de evitar danos ao paciente como também o de aumentar a chance de sucesso na terapia. Esses *softwares* são chamados de Sistemas de Planejamento de Tratamento (12, 14, 39).

Basicamente os sistemas de planejamento efetuam o cálculo do transporte de radiação de duas formas, ou por intermédio de modelos determinísticos, através da solução algébrica das equações de transporte de radiação, ou mediante modelos estocásticos usando o Método de Monte Carlo para realizar o transporte de radiação. Os modelos determinísticos têm a desvantagem de serem menos precisos em relação aos estocásticos, porém eles possuem a vantagem de serem muitos mais rápidos no processamento computacional. Devido a estas características foram os mais empregados inicialmente (12, 39). Atualmente devido aos avanços tecnológicos, na área da informática e ciência da computação, houve um aumento considerável de sistemas de planejamento utilizando códigos de Monte Carlo a fim efetuar os cálculos de transporte de radiação (40, 39).

Esse aumento pode ser comprovado pelo número de publicações disponíveis no banco de dados do PubMed[1]. Com isso, foi realizada uma pesquisa no banco de dados do PubMed com os seguintes termos de busca: "*Monte Carlo*" AND "*Radiotherapy*" e "*Monte Carlo*" AND *Nuclear Medicine* OR *Radiopharmaceuticals*. Os resultados dessas buscas são apresentados na Figura 3.1.

[1] <http://www.ncbi.nlm.nih.gov/pubmed>

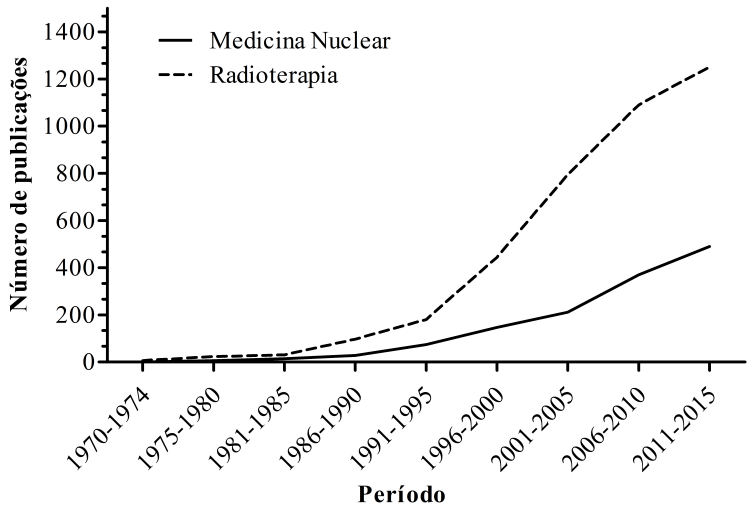

Figura 3.1: Número de artigos que possuem o termo "*Monte Carlo*" em publicações com os termos "*Radiotherapy*" e "*Nuclear Medicine*" ou "*Radiopharmaceuticals*".

Observa-se, na Figura 3.1, que há um avanço muito maior no uso de códigos de Monte Carlo na radioterapia do que na medicina nuclear a fim de estimar a distribuição de dose nos paciente.

Para averiguar tal situação foi feita uma busca na base de dados do PubMed, primeiramente com os termos "*Therapy*" AND "*Radiopharmaceuticals*" e depois com os termos "*Dosimetry*" and "*Radiopharmaceuticals*". A Figura 3.2 mostra o resultado dessa busca.

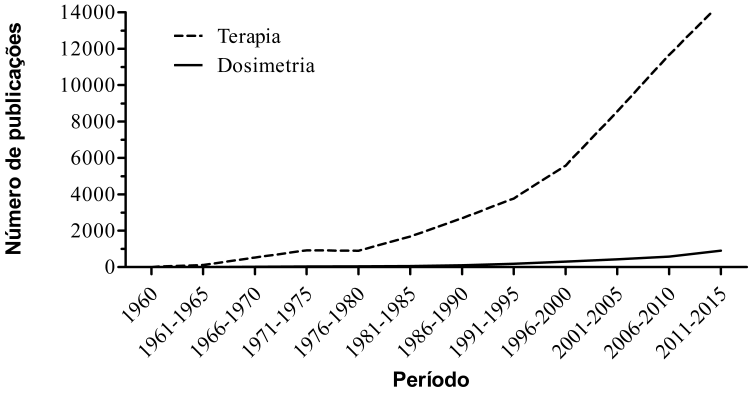

Figura 3.2: Número de artigos que possuem o termo "*Radiopharmaceuticals*" em publicações com os termos "*Therapy*" e "*Dosimetry*".

É possível observar, através do gráfico da Figura 3.2, que muitas publicações em medicina nuclear não abordam aspectos dosimétricos. Isso demostra que muitas publicações que abordam sobre a terapia em medicina nuclear não consideram a dose absorvida de

radiação a fim de verificar sua eficácia no tratamento (41).

Outro fator importante a ser considerado é a administração de quantidades semelhantes de radiofármacos para diferentes pacientes nos procedimentos terapêuticos em medicina nuclear (41, 42, 43).

O motivo do número reduzido de planejamento em medicina nuclear, se da devido a quatro fatores (41).

1. Devido ao fator histórico desse tipo de terapia ser estabelecido de forma empírica;
2. Aproximadamente 85% dos casos possuem uma eficiência terapêutica satisfatória;
3. O planejamento requer de quatro a cincos dias para ser realizado;
4. Falta de métodos padronizados e pessoal capacitado em dosimetria interna.

Visando a solução desses problemas na área de medicina nuclear a Agência Internacional de Energia Atômica (IAEA) tem incentivado mundialmente a implantação de planejamentos em medicina nuclear. No Brasil a terapia individualizada de paciente ainda é pouco expressiva. A implantação do planejamento em medicina nuclear poderia propiciar um melhor atendimento ao paciente e em alguns casos até reduzir custos terapêuticos, como por exemplo, a diminuição da atividade administrada e a redução do número de retratamentos (41).

A seguir serão descritos com maior detalhes os sistemas de planejamento de tratamentos que utilizam códigos de Monte Carlo nas áreas de medicina nuclear e radioterapia.

3.4.1 Sistemas de planejamento do tratamento com códigos de Monte Carlo em Medicina Nuclear

A dosimetria em medicina nuclear mediante códigos de Monte Carlo possui basicamente as seguintes etapas (42, 43).

Primeiramente, há a necessidade de se caracterizar a geometria do paciente. Para isso, são adquiridas as imagens de Tomografia Computadorizada que fornecem tanto a geometria como também as características dos tecidos do paciente que são as composições e densidades dos tecidos. Essas informações são adquiridas para cada *voxel* da imagem (39).

Em segundo lugar é necessário a caracterização da fonte de radiação e para isso são utilizadas imagens de SPECT/PET que fornecem a distribuição da atividade interna ao paciente (42, 43).

São necessárias mais de uma imagem de SPECT/PET para definir a distribuição da atividade no organismo. Isto porque, é necessário calcular a atividade acumulada nos órgão de interesse mediante essas imagens. A atividade acumulada é definida como o produto da atividade do radionuclídeo, presente no órgão de interesse, pelo tempo em que ele permanece no órgão. Assim, mediante uma sequência de imagens é possível estimar a atividade acumulada. (42, 43, 39).

Essas imagens, devido à atenuação da radiação com a matéria, também, necessitam ser previamente corrigidas a fim de fornecerem de maneira mais precisa a distribuição de atividade (39).

A terceira e epata é transmitir estas informações aos códigos de Monte Carlo de forma que eles possam efetuar o cálculo de dose (39). Por fim, as informações provenientes dos códigos de Monte Carlo são interpretadas a fim de construir o mapa da distribuição de dose no paciente.

3.4.1.1 Softwares utilizados em Sistema de Planejamento com códigos de Monte Carlo

A Tabela 3.1 mostra um lista de softwares utilizados para cálculo de dose que são baseados em códigos de Monte Carlo para medicina nuclear.

Tabela 3.1 - Softwares baseados em métodos de Monte Carlo para cálculo de dose em Medicina Nuclear

Software	Descrição
VIDA	Sistema de Planejamento em terapia radionuclídica baseado no código Geant4 (44).
AMIGO	Software para cálculo dosimétrico em Medicina Nuclear baseado no código MCNP (6, 45).
MINERVA	Sistema de Planejamento em terapia radionuclídica (46).
GATE	Ferramenta de simulação para PET e SPECT (47, 48, 49).
OEDIPE	Sistema de planejamento em terapia radionuclídica baseado no código MCNPX (50).
VoxelDose	Sistema de planejamento em terapia radionuclídica baseado no código EGS4 (51).
Dose3D	Software para dosimetria interna baseado no código EGS4 (52).

Continuação da Tabela 3.1

Software	Descrição
RMDP	Sistema para quantificação, registro e dosimetria de imagens ^{131}I SPECT (53).

3.4.2 *Sistemas de planejamento do tratamento com códigos de Monte Carlo em Radioterapia*

Com o objetivo de simular um tratamento de teleterapia com os códigos de Monte Carlo são necessárias as seguintes informações: a caracterização da fonte de radiação, a definição da geometria do paciente e as informações pertinentes aos códigos de Monte Carlo como a utilização de técnicas de redução de variância a fim de aumentar a eficiência dos resultados.

3.4.2.1 *Caracterização da fonte de radiação*

Simulação do cabeçote do acelerador linear: A primeira informação que será abordada é a caracterização da fonte de radiação em sistemas de planejamento em teleterapia. Isso é realizado através da modelagem do feixe de radiação emitido pelos equipamentos de teleterapia (54).

Na Figura 3.3 (54) é representado o esquema de um o cabeçote de acelerador linear (54). Na produção de um feixe de fótons, um feixe de elétrons incide em um alvo de tungstênio fazendo com que haja emissão de fótons pelo efeito de *bremsstrahlung*. Esse feixe de fótons, passa por colimadores e filtros que possuem o objetivo de transformar o feixe em um campo uniforme. Há uma câmera de ionização colocada com o intuito de analisar o feixe produzido pelo acelerador linear e há também um espelho onde ocorre a incidência de luz visível para que o radioterapeuta consiga visualizar o campo de radiação. Após o espelho, há o colimador *multileaf* que é uma combinação de finas folhas de comiladores as quais podem ser posicionadas independentemente uma das outras. Isto permite produzir diferentes formas de campos de radiação (54).

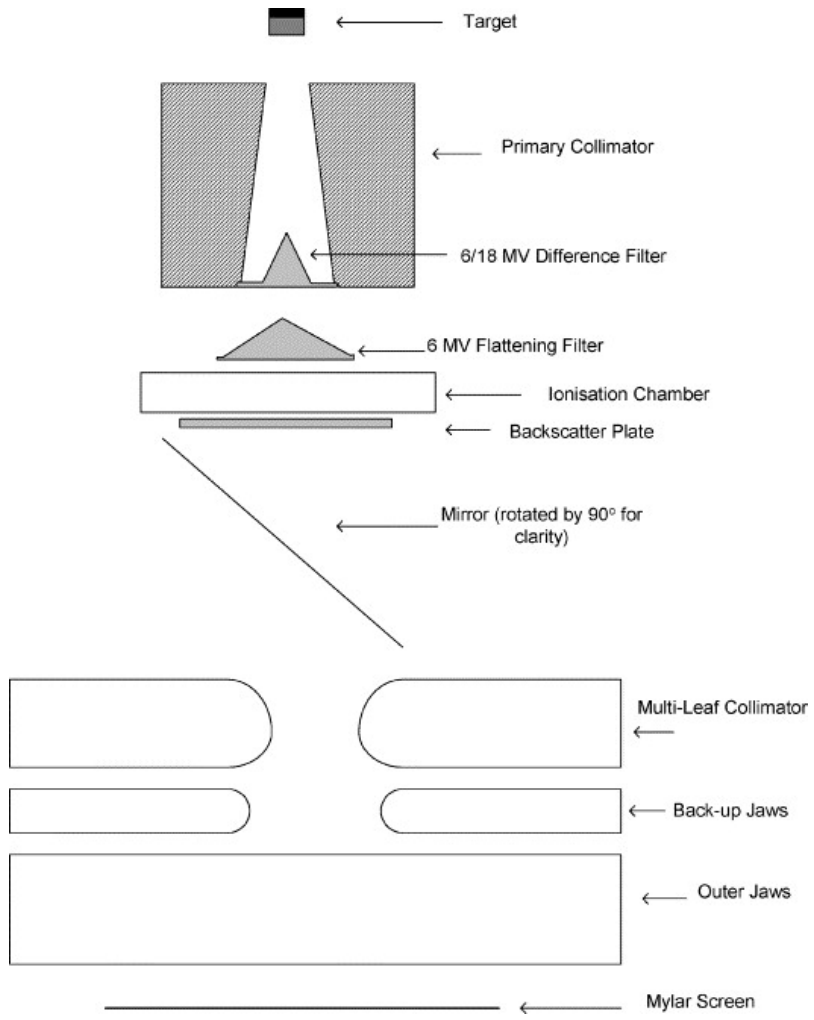

Figura 3.3: Esquema do cabeçote de um acelerador linear Elekta SLi plus.

Um dos problemas de se modelar o cabeçote é o alto tempo computacional despendido nessa tarefa. Com isso, uma das soluções é modelar o feixe de radiação em duas partes. A primeira parte consiste na modelagem dos componentes estáticos do cabeçote, isto é, a região superior do cabeçote antes de chegar nos colimadores *multileaf*. E a segunda parte é modelar o feixe desde os colimadores *multileaf* até o paciente (54).

Dessa forma, a parte de cima pode ser modelada uma única vez com a geração de um arquivo denominado espaço de fase. Este arquivo contem todos os parâmetros necessários para o transporte de partículas (tipo de partícula, coordenadas, cossenos diretores, energia e peso das partículas). Por fim, o espaço de fase é então utilizado como dado de entrada na simulação da segunda parte (54).

A qualidade do feixe de radiação simulado depende fortemente de todas as características físicas mencionadas (54).

Simulação das fontes de braquiterapia: Códigos que utilizamo Método de Monte Carlo são ferramentas dosimétricas essenciais para a braquiterapia, tanto na parte clínica como na pesquisa. Um de seus usos mais comuns são os cálculos da distribuição da taxa de dose em volta das fontes de radiação (55).

As fonte de braquiterapia podem conter radionuclídeos de baixa energia (^{103}Pd, ^{125}I, ^{131}I) ou radionuclídeos de alta energia (^{192}Ir, ^{137}Cs) (55).

Os radionuclídeos de baixa energia são utilizados em braquiterapia de baixa taxa de dose (LDR - *Low Dose Rate*) em que o radionuclídeo, juntamente com um marcador radio-opaco, são encapsulados na forma de sementes as quais serão colocadas próximas da região tumoral (55).

Os radionuclídeos de alta energia são utilizados em braquiterapia de alta taxa de dose (HDR - *High Dose Rate*) que utilizam técnicas denominados "*remote afterloader*" a fim levar as cápsulas contendo fontes de alta energia até a região tumoral (55).

Para que os tratamentos possam ser simulados nos códigos de Monte Carlo, há a necessidade de se construírem essas fontes de radiação (55).

A fim de simular as sementes de braquiterapia são necessários não somente os dados detalhados da geometria da semente como também de todos os componentes próximos a ela (55, 56).

Em muitos casos as informações da fonte e de todas as estruturas envolvidas em sua construção são adquiridas atráves de dados fornecidos pelo fabricante. Porém, só isso não é o suficiente. Há também a necessidade de se validar com medidas experimentais (55, 56).

Devido à grande complexidade na modelagem das fontes de radiação e também ao tempo computacional despendido nessa tarefa são gerados os espaços de fase para as fontes de braquiterapia. O arquivo do espaço de fase é então utilizado futuramente na simulação do tratamento (55, 56, 57).

Para simular as sementes localizadas dentro do paciente é preciso obter informações de seus posicionamentos. Tais posições podem ser adquiridas através de imagens de Tomografia Computadorizada ou Ressonância Magnética (58, 55).

3.4.2.2 Fatores geométricos do tratamento

A segunda informação que será abordada é a modelagem do paciente dentro dos códigos de Monte Carlo. Para isso são utilizados as informações adquiridas da Tomografia

Computadorizada (CT) a fim de converter em parâmetros do tecido (54).

Para realizar essas conversões os tomógrafos devem ser previamente calibrados. Inicialmente, a maioria dos valores extraídos dos tomógrafos eram em relação à densidade eletrônica, devido a uma correlação bem estabelecida entre os os valores adquiridos pelos tomógrafos com a densidade eletrônica (59).

No entanto, é mais importante para os códigos de Monte Carlo a densidade de massa dos tecidos do que a densidade eletrônica do material (54). Outro fator que também é muito importante para os códigos de Monte Carlo, são as composições dos tecidos. Com isso surgiram diversos trabalhos a fim de calibrar os Tomógrafos em relação à densidade de massa como também as composições dos tecidos (60, 61).

3.4.2.3 Informações pertinentes aos códigos de Monte Carlo.

Técnicas de redução de variância: Para obter uma eficiência significativa em seus resultados os códigos de Monte Carlo requerem muito tempo computacional. Para aumentar a eficiência sem prejudicar o tempo computacional são utilizadas técnicas de redução de variância (55).

A eficiência η é definida com $\eta = 1/\sigma^2 T$, onde σ^2 é a variância, que é proporcional ao inverso do número de histórias[2], e T é o tempo de processamento consumido para processar N histórias (55).

Assim, há basicamente duas formas de se aumentar a eficiência. Uma é diminuindo a variância do problema mediante métodos estatísticos. A outra é reduzir a quantidade do tempo gasto por história (55).

Dessa forma, a maioria dos softwares baseados em códigos de Monte Carlo utilizam técnicas de redução de variância (62).

Malha virtual: Muitos dos sistemas de planejamento que utilizam códigos de Monte Carlo estimam a dose para cada *voxel*. As dimensões desses *voxels* são especificadas pelas Tomografias Computadorizadas (CT). Essa abordagem dependendo da quantidade de *voxels* pode causar um inaceitável uso de memória para armazenar os dados da dose, a menos que a resolução do CT seja diminuída para limitar o número de *voxels*. Uma das alternativas, a fim de solucionar esse problema, é criar uma grade virtual sobreposta aos

[2] A história é o percurso de uma partícula no sistema.

voxels com a finalidade de armazenar os dados da dose (54).

Conversão de dose para dose na água: Geralmente sistemas de cálculo de dose com códigos de Monte Carlo determinam a dose para o meio no qual o transporte é simulado (54, 55, 57).

Porém os cálculos de dose convencionais tanto em teleterapia como braquiterapia determinam a dose na água (54, 55, 57).

Dessa forma é necessário a conversão dos resultados de dose obtidos pelos códigos de Monte Carlo para dose na água (54, 55, 57).

3.4.2.4 Softwares utilizados em Sistema de Planejamento com códigos de Monte Carlo

A Tabela 3.2 mostra um lista de softwares utilizados para cálculo de dose que são baseados em códigos de Monte Carlo.

Tabela 3.2 - Softwares baseados em códigos de Monte Carlo para Radioterapia

Software	Descrição
bGPUMCD	Algorítmo de Monte Carlo executado em Unidade de Processamento Gráfico (GPU) para rápido cálculo de dose em implante de próstata permanente (63).
AMIGOBrachy	Sistema de planejamento e braquiterapia com uso do código de Monte Carlo MCNP6 (64).
ALGEBRA	Algorítmo para dosimetria heterogênea baseado no Geant4 para braquiterapia (65).
BrachyDose	Sistema de cálculo de dose baseado no EGSnrc e desenvolvido para casos de implante de prostata (66).
PEREGRINE	Sistema de cálculo de dose tridimensional baseado em Monte Carlo desenvolvido para Radioterapia (67).
DPM	Simulação do transporte de elétrons e fótons para radioterapia (68).
MCDOSE	Ferramenta de cálculo de dose para sistemas de planejamento de radioterapia (69).
MCPI	Cálculo de dose pelo método de Monte Carlo para implante de próstata com ^{125}I e ^{103}Pd (70).
VMC, XVMC, VMC++	Simulação de transporte de elétrons e fótons (71, 72, 73).
ORANGE	Sistema de Cálculo de dose para Radioterapia (74, 75) .

3.5 Estrutura de sistemas modulares de planejamento baseados em códigos de Monte Carlo.

A seguir serão descritas as estruturas de três sistemas modulares altamente flexíveis. O entendimento dessas estruturas contribuíram para a construção do sistema proposto nesse trabalho.

Voxel2MCNP: O software *Voxel2MCNP* (10) permite a modelagem de cenários de proteção radiológica com o código *MCNPX*. Umas das principais questões abordas pelo artigo (10) é sobre a diversidade de informações que são necessárias para se conseguir efetuar a modelagem de um cenário de proteção radiológica mediante códigos de Monte Carlo. Tal ponto é importantíssimo para o presente trabalho pois este enfrentou o mesmo problema, isto é, fornecer uma maneira de interpretar diferentes fontes de dados com a finalidade de gerar o arquivo de configuração para os códigos de Monte de Carlo.

Há diversos softwares que podem auxiliar na construção dos itens necessários para construção dos arquivos de configuração dos códigos de Monte Carlo, porém, muitos deles não são compatíveis com os códigos de Monte Carlo. Com isso, os usuários são forçados a criarem diversos códigos para interpretar a saída desses softwares a fim de construírem o cenário desejado. Esse tipo de tarefa consome muito tempo do usuário além de diminuir a reprodutibilidade dos processo de modelagem que consequentemente diminui a chance da comparação entre resultados (10).

Com o objetivo de solucionar este problema Pölz et al.(10) desenvolveram um formato de arquivo em XML (*eXtensible Markup Language*), estabelecido com *V2M XML scheme* que engloba todo o cenário de proteção radiológica. Assim a comunicação de dados e os esforços na integração com os códigos de Monte Carlo são facilitados. Com base nisso, o software *Voxel2MCNP* apresenta um sistema de *plugins* [3] que possuem a capacidade de exportar e importar dados de sistemas externos utilizando o padrão estabelecido no *V2M XML scheme*. Através desse padrão é criado o arquivo de configuração do código de Monte Carlo *MCNPX*.

A grande desvantagem desse sistema é o uso do formato padrão *V2M XML scheme*, pois caso haja alguma alteração nesse padrão todos os sistemas que já foram integrados

[3] Plugin, ou também denominado módulo de extensão, é uma pequena aplicação que fornece novas funcionalidades, muito específicas, para uma aplicação maior.

ao *Voxel2MCNP* serão prejudicados.

MINERVA: O software MINERVA ("*Modality INclusive Environment for Radiotherapeutic Variable Analysis*") é um *framework* [4] para cálculos dosimétricos, a fim de ser utilizado em uma variedade de tratamentos de radioterapia (46).

O sistema foi construído de maneira modular permitindo que seja extendido por meio da criação de *plugins* com o uso de uma API [5] ("*Application Programming Interface*") (46).

Com a finalidade de adquirir os dados para a simulação, ele foi composto por quatro módulos principais. O primeiro é o da Imagem, responsável por importar dados de imagens em diferentes formatos. O segundo é o do Modelo, responsável por construir um modelo do paciente baseado nas imagens. O terceiro é o módulo da Análise, que permite configurar dados específicos ao tratamento. O último módulo é o do Paciente o qual recebe todas essas informações e as processa (46).

A API do MINERVA recebe os dados processados do módulo do Paciente e repassá-os aos *plugins* referente à construção da fonte de radiação e ao método para transporte de radiação. Todo esse processo torna o sistema altamente flexível permitindo-lhe abordar diferentes tipos de tratamentos (46).

A desvantagem desse sistema é justamente a API, pois caso ela seja alterada todos os plugins já feitos serão prejudicados.

GATE: O GATE ("*Geant4 Application for Tomographic Emission*") é um sistema baseado no código Geant4, escrito de maneira modular e que permite a simulação de sistemas em medicina nuclear através de *scripts* feitos por uma linguagem própria (47, 76).

O sistema preconiza alguns pontos interessantes. Primeiramente o usuário final não precisa ter conhecimento da linguagem a qual o Geant4 impõem, isto porque, o Geant4 é escrito em C++ e exige que o usário saiba a linguagem para assim poder simular algo. Outro aspecto é que o sistema seja extensível, isto é, possibilite a adição de novas funcionalidades sem prejudicar as existentes (47, 76).

O sistema é formado por três camadas. A primeira camada é o núcleo, "*Core*", baseado no Geant4 e onde são definidas as principais ferramentas e características do GATE. A

[4] O *framework* fornece um conjunto de ferramentas a fim de se solucionar um família de problemas semelhantes.

[5] A API fornece informações de um determinado sistema por meio de interfaces pré definidas.

segunda camada é a aplicação, "*Application*", onde há um conjunto de estruturas de classes extensíveis baseadas na camada do núcleo. A terceira camada é a usuário, "*User*", na qual os usuários podem simular experimentos mediante uma versão extendida de uma linguagem de *scripts* do Geant4 (47, 76).

Por intermédio dessa estrutura de camadas é possível não somente simular casos particulares de medicina nuclear como também casos de braquiterapia, radioterapia e até cálculo de dose por equipamentos de Tomografia Computadorizada (47, 76).

As principais devantagens desse sistema são incluir informações na camada do usuário, que requerirá mudanças na camada da aplicação, e incluir informações na camada da aplicação a qual provocará mudanças na camada do usuário.

Capítulo 4

Materiais e Métodos

Neste capítulo serão apresentadas os materiais e métodos utilizados neste trabalho.

4.1 Materiais

4.1.1 MCNP6 - Monte Carlo N-Particle Transport Code

O código de transporte de radiação MCNP, denominado *Monte Carlo N-Particle Transport Code*, permite a simulação tridimensional de 37 tipos diferentes de partículas para diversos fins, como por exemplo, cálculo da criticalidade em reatores nucleares, cálculo de blindagem, dosimetria entre outros fins (77).

A nova versão desse código é o MCNP6 que incorporou as versões, MCNP5 e MCNPX. O MCNP possui diversos arquivos a fim de realizar a simulação sendo que um deles é especificado pelo usuário, denominado arquivo *INP*. Dentro desse arquivo deverão ser descritos todos os parâmetros necessários para a realização da simulação. No Apêndice A é apresentada a estrutura do arquivo *INP*.

Dentre os diversos trabalhos, na área de física médica que utilizam o MCNP, encontram-se os cálculos de dosimetria aplicado à cálculos de dose em áreas como radioterapia, braquiterapia, radioimunoterapia, entre outras (64, 10, 78, 79).

4.1.2 AMIGO (Algorithm for Medical Image Generating Object)

O *AMIGO* é a nova versão do antigo SCMS (*Segmented Construction Mannequins Software*) (6), e reescrito em Java (45). Tem a finalidade de servir como um *software* de interface entre as imagens médicas e o código MCNP. Suas duas versões foram desenvolvidas no Centro de Engenharia Nuclear do IPEN.

Com ele é possível criar arquivos do MCNP referente à casos de Medicina Nuclear e por este motivo ele foi utilizado neste trabalho.

4.1.3 *AMIGOBrachy A Medical Image-based Graphical platfOrmd - Brachytherapy module*

O *AMIGOBrachy* é um software desenvolvido para a área de Braquiterapia afim de possibilitar a simulação de planejamento de tratamento clínico com códigos de Monte Carlo. Para isso ele possui uma interface gráfica extremamente amigável com recursos de processamento de imagens médicas, importação e edição de arquivos de sistemas de planejamentos de tratamentos além de permitir configurar parâmetros de simulação e analisar os resultados (64).

Neste trabalho o *AMIGOBrachy* foi utilizado como referência para sistemas de planejamento com uso do código MCNP em Braquiterapia.

4.1.4 *ICCT - Image Converter Computed Tomography*

O *ICCT* é um *software* que cria objetos simuladores em *voxels* de qualquer paciente, mediante imagens de Tomografia Computadorizada (CT).

Após uma prévia calibração com o Tomógrafo que será utilizado no imageamento, o *ICCT* gera um mapa de 24 grupos de tecidos com seus respectivos intervalos na escala de *Hounsfield*, específicos para esse tomógrafo. Porém diferentemente de muitos métodos, os quais utilizam densidades e composições fixas, ele extrai esses parâmetros dos tecidos do paciente através das imagens de CT.

Ou seja, ele é totalmente dependente do CT, isto é, para que o objeto em *voxel* seja representado de maneira mais real possível, o CT utilizado na calibração deve ser o mesmo utilizado para fazer as imagens do paciente. É através do mapa dos tecidos que é realizada a segmentação das imagens tomográficas do paciente com a finalidade da criação do objeto simulador (80, 8).

4.1.5 *Objeto Simulador Antropomórfico RANDO® Phantoms*

O objeto simulador *RANDO® Phantoms* é semelhante ao objetos simuladores seccionais. Ele foi construído com esqueleto humano fundido ao material que é radiológicamente

equivalente ao tecido mole. Ele permite o mapeamento da distribuição de dose detalhado que é essencial para avaliar planejamentos de tratamentos de Radioterapia (81).

Ele foi utilizado neste trabalho para validar a integração do *ICCT* com o IBMC que será discutida na seção 5.1.4.

4.1.6 Java

JAVA é uma linguagem de programação resultante de um projeto desenvolvido em 1991 por um grupo de engenheiros da *Sun Microsystem*. Inicialmente, esses engenheiros tinham a missão de criar uma linguagem que seria utilizada em diversos aparelhos, tais como fornos e telefones. Além disso, essa linguagem deveria permitir a comunicação entre os diversos aparelhos de uma casa, mediante a execução de pequenos códigos, presentes nesses aparelhos (82).

Os engenheiros responsáveis pelo projeto optaram por utilizar a linguagem de programação C++ em conjunto a um conceito de máquina virtual a qual possui a finalidade de executar programas da mesma forma que um computador real. A opção por essa linguagem deve-se ao fato dela ser muito poderosa e ter a vantagem de ser orientada a objetos. Quanto ao conceito de máquinas virtuais, foi à solução encontrada para a execução em diversas arquiteturas (82).

Como mencionado, à linguagem seria desenvolvida para ambientes residenciais com o intuito de permitir que aparelhos eletrodomésticos fossem capazes de se comunicar entre si. Porém, atualmente tornou-se uma das linguagens mais conhecidas e utilizadas do mundo (82).

O JAVA possui uma importante característica que é a de ser uma linguagem orientada a objetos. Softwares que se baseiam na orientação a objetos têm a vantagem de serem facilmente alterados em algum momento posterior, para uma melhor adaptação às necessidades apresentadas.

4.1.7 Conceitos de Orientação a Objetos

A programação orientada a objetos se utiliza de um conjunto de entidades denominadas "*Classes*" e "*objetos*" a fim de permitir a estruturação dos dados e conceitos do *software*.

4.1.7.1 Classes e Objetos

Na programação orientada a objetos além dos tipos primitivos, como inteiros, ponto flutuante, vetor unidimensional entre outros é possível criar novos tipos por intermédio da criação de classes, sendo que esses novos tipos podem ser instanciados a fim de criar os objetos. Dessa forma cada objeto criado corresponde à uma classe em específico. Em termos gerais os objetos representam algo concreto enquanto as classe representam uma abstração à esses objetos.

As classes possuem atributos e métodos que representam seus estados e comportamentos, respectivamente. Ao serem instanciados os objetos preenchem os valores referente aos atributos da classe e executam seus métodos. A construção de um sistema orientado a objetos consiste em definir um conjunto de classes e as relações entre elas a fim de solucionar um determinado problema.

4.1.7.2 Herança

A herança permite que as classes herdem características de outras classes, isto é, possam herdar os atributos e métodos de outra classe. Um exemplo é a classe "Cachorro" que tem suas características como por exemplo latir e comer, e também que apresenta atributos como nome, sexo e idade, herdar características da classe "Animal" que possui por exemplo a característica de animal doméstico. Assim como é dito que cachorro é um animal também é dito que tal classe é do tipo referente à outra classe. Por exemplo, a classe "Cachorro" e do tipo "Animal". Isso permite que outras classes se beneficiem de comportamentos comuns entre elas sem duplicar código.

4.1.7.3 Polimorfismo

O polimorfismo ocorre quando a herança é utilizada. Ao dizer que uma determinada Classe "A" herda a Classe "B", permite tratar a Classe "A" como se fosse a "B". Dessa forma o polimorfismo permite que um objeto possa ser visto como qualquer uma de suas classes.

4.1.8 Padrões de Projetos

Desenvolver um sistema orientado a objetos, não é uma tarefa simples. Caso esse sistema tenha que ser reutilizável, a complexidade aumenta drasticamente. Isto, por que

esse sistema deve ser específico, para resolver um determinado problema, ser flexível, a fim de atender as especificações futuras e ser reutilizável, o que o permitiria ser aproveitado em outros sistemas (83).

Uma das ferramentas mais importantes no desenvolvimento de tais sistemas são os Padrões de Projetos ("*Desing Patterns*"). Esses padrões, de forma resumida, são diagramações da interação entre classes e objetos, que tem como propósito a solução de problemas recorrentes em sistemas orientados a objetos.

Devido ao fato do poder em abstrair tais problemas, os Padrões de Projetos também possuem um papel importante na melhora da comunicação entre os desenvolvedores, não só de dentro do projeto como também externos a ele. Esse aspecto ajuda no processo de reutilização do sistema (83).

4.1.8.1 Padrão - Visitor

Segundo Gamma et al.(83), "O padrão *Visitor* representa uma operação a ser executada sobre os elementos da estrutura de um objeto. O vistor permite que você defina uma nova operação sem mudar as classes dos elementos sobre os quais opera."

Para exemplificar, suponha que exista uma classe que seja abstração das principais informações de um empresa, denominada "*Relatorio*". Dentro dessa classe haverá diversos atributos e métodos referente a este assunto. Suponha então que se deseja fazer diversos relatórios com a manipulação desses dados. Para fazer isso, poderia-se criar métodos na Classe "*Relatorio*" a fim de gerar os diversos relatórios desejados. Porém esta é uma abordagem ruim pois a cada novo tipo de relatório o desenvolvedor teria que implementar um novo método na classe pré existente "*Relatorio*", o que poderia gerar grande dificuldade de manutenção do código e prejudicar os relatórios anteriormente criados. O padrão *Visitor* pode ser utilizado nesse caso para flexibilizar e facilitar a construção de novos relatórios.

Para resolver esse problema com o padrão *Visitor* é criada uma Classe contendo a abstração do que seria necessário conter no relatório, por exemplo, título, tabelas, entre outros métodos, sem conter a real implementação, são os denominados métodos abstratos. Mediante essa classe são criadas outras que herdam essa classe e implementam os métodos abstratos. Essas Classes irão produzir os diferentes tipos de relatórios, nessa abordagem a criação de relatórios se flexibiliza pois a cada novo relatório somente é preciso criar uma nova classe sem modificar as existentes.

Esse padrão será utilizado no projeto para flexibilizar a criação de novos sistemas de planejamento dosimétrico no MCNP.

4.2 Métodos

4.2.1 Modelo genérico de um Sistema de Planejamento Dosimétrico que utiliza o código MCNP

Os sistemas de planejamento dosimétrico que utilizam o código *MCNP* necessitam basicamente de duas informações para simular um determinado tratamento de Radioterapia ou Medicina Nuclear. A primeira informação seria a geometria do paciente e a segunda, a fonte de radiação (42, 55, 6).

Além dessas informações, os sistemas de planejamento precisam especificar alguns parâmetros correspondentes às configurações do *MCNP*, como por exemplo, os resultados esperados após à simulação, especificados pelos *tally*, ou declarar técnicas de redução de variância (55, 6).

Após a coleta dessas informações, os sistemas de planejamento constroem o arquivo de entrada, *INP*, do *MCNP* cuja especificação é apresentada no apêndice A (55, 6).

A fim de representar estas etapas foi elaborada a Figura 4.1, a qual contêm a representação de um modelo genérico referente aos sistemas de planejamentos com o uso do código de Monte Carlo *MCNP* (6).

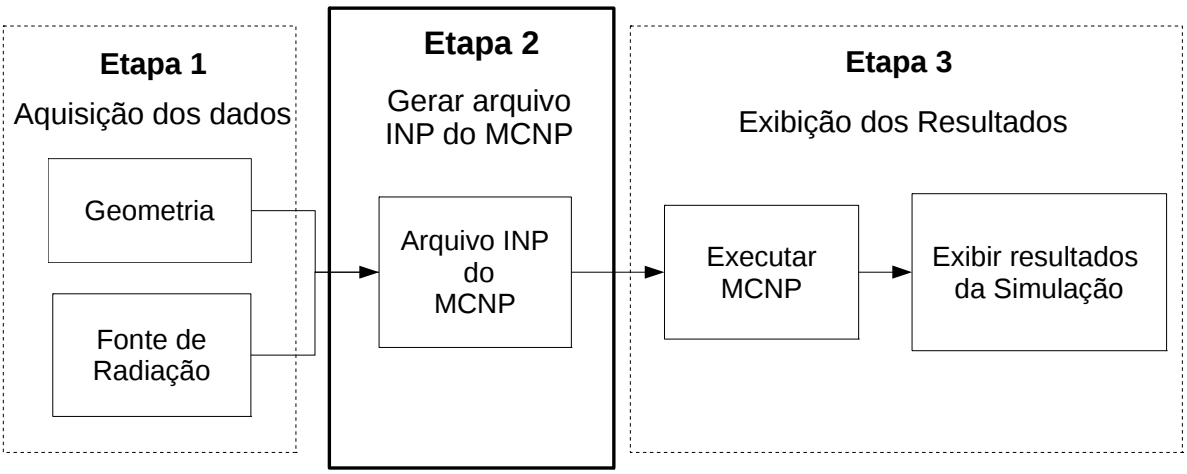

Figura 4.1: Modelo genérico dos Sistemas de Planejamentos com o uso do código de Monte Carlo MCNP.

Na primeira etapa, presente na Figura 4.1, ocorrem as aquisições dos dados do trata-

mento sendo que cada tipo de tratamento poderá fornecer esses dados de maneira distintas. Por exemplo, as aquisições dos dados da geometria do paciente, em Medicina Nuclear, podem ser fornecidas por imagens de Tomografia Computadorizada (CT) ou através de objetos simuladores antropomórficos e os dados referente à fonte de radiação através das imagens de SPECT (*Single Photon Emission Computed Tomography*) ou PET (*Positron Emission Tomography*) (6).

Em Radioterapia, os dados da geometria poderiam ser adquiridos de maneira similar aos do tratamento anterior, porém os dados da fonte seriam especificados diferentemente, sendo uma das opções a simulação do cabeçote do acelerador ou a utilização de arquivos de espaço de fase que contêm as informações da fonte de radiação provindas do cabeçote (55).

Sistemas que permitem o gerenciamento das diferentes representações dos dados relativos à primeira etapa deverão ser altamente flexíveis a fim de contemplar diferentes terapias. Essa maleabilidade do sistema é ainda mais importante ao se contemplar tratamentos futuros, onde a caracterização da fonte e geometria ainda são desconhecidas (10, 46, 47).

Na segunda etapa ocorrem os processamentos das informações provenientes da primeira etapa a fim de construir o arquivo *INP* do *MCNP*. Com isso, esta etapa torna-se altamente complexa, devido às diversidades dos dados provenientes da primeira etapa juntamente com o fato dos arquivos *INP* gerados serem diferentes para cada tratamento. Isto é, o sistema tem que permitir que um conjunto de dados adquiridos na primeira etapa possam ser utilizados na construção de diferentes arquivos *INP*. Por exemplo, os dados da geometria em Medicina Nuclear, como já mencionado, são frequentemente provenientes das imagens de Tomografia Computadorizada, assim como nos tratamentos de Radioterapia. Dessa forma, fica evidente que os dados provenientes do CT podem ser utilizados tanto em casos de Medicina Nuclear como em Radioterapia.

Já na terceira e última etapa ocorrem as interpretações dos resultados gerados pela simulação do tratamento no *MCNP* e a apresentação dos resultados, de maneira amigável, ao usuário final.

4.2.2 IBMC (Input Builder of Monte Carlo Code): Constrói arquivos de entrada para códigos de Monte Carlo

O IBMC foi concebido para auxiliar pesquisadores de todo o mundo na elaboração, de maneira rápida e prática, dos arquivos de entrada para determinados códigos de Monte

Carlo que assim o exigem, entre eles o *MCNP*. Ele foi desenvolvido, com base no modelo genérico apresentado na seção 4.2.1.

Ele é um software de código fonte aberto, com a licença MIT[1], orientado a objetos e desenvolvido na linguagem de Programação Java. O seu código está disponível no repositório do *Bitbucket*[2] juntamente com sua documentação[3].

Primeiramente tem-se que o IBMC, supõem a existência de softwares que deem suporte à primeira e última etapa do modelo apresentado na Figura 4.1. Isto é, ele não contemplará ferramentas, por exemplo, de processamento de imagens, assim como ferramentas específicas para cada tipo de tratamento, como o espectro de radiação para fontes de Medicina Nuclear, por exemplo. Esses dados deverão ser previamente processados pelo desenvolvedor, que utilizar o IBMC, seja por meio de *scripts*, para buscar estas informações diretamente nos softwares de origem, ou mediante arquivos textos com as informações pré cadastradas. Enfim, há diversas maneiras de se adquirir os dados provenientes dos softwares terceiros.

A principal funcionalidade do IBMC é auxiliar a construção de sistemas que integrem a primeira e terceira etapas. Assim, com base no IBMC, será possível desenvolver diversos sistemas de planejamento que utilizem os códigos de Monte Carlo. O processo preconizado pelo IBMC é apresentado na Figura 4.2.

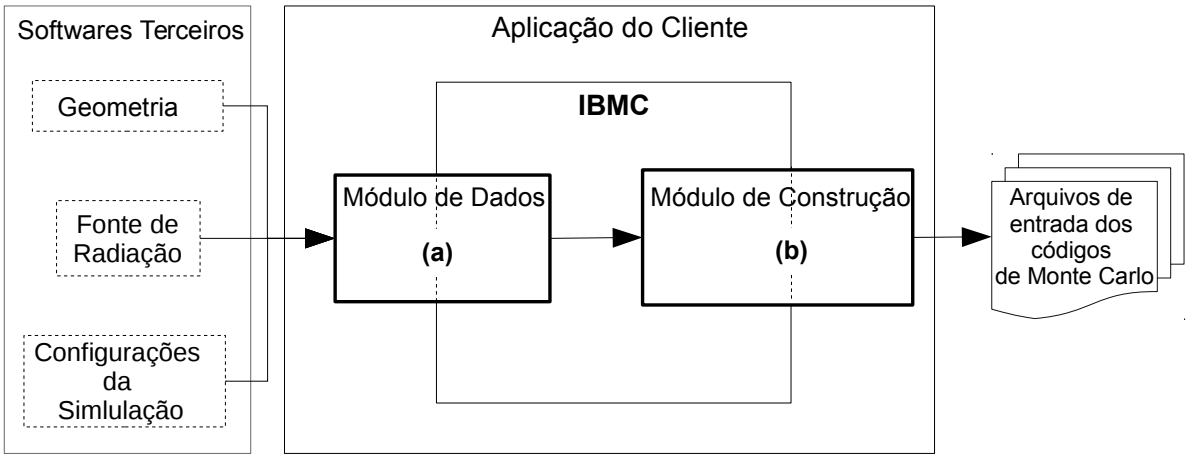

Figura 4.2: Modelo do fluxo dos dados estabelecido pelo IBMC.

O IBMC apresenta dois módulos principais, o **Módulo de Dados** que está relacionado

[1] <http://opensource.org/licenses/MIT>
[2] <https://bitbucket.org/fmassicano/ibmc-input-builder-of-monte-carlo>
[3] <https://bytebucket.org/fmassicano/ibmc-input-builder-of-monte-carlo/raw/697899aff2b84f307dc4ceff3e12c51e3bd6f04e/Documentation/refman.pdf?token=aefc85977aecc73f768538edf2bfba8dbc6862fc>

com a aquisição dos dados através dos *softwares* terceiros, Figura 4.2(a), e o **Módulo de Construção** relacionado com a construção dos arquivos de entrada para os códigos de Monte Carlo, Figura 4.2(b). Ambos os módulos possuem relações com o IBMC e com a aplicação do cliente, isto é, há informações que competem ao IBMC realizar e outras que é de responsabilidade da aplicação do cliente.

Para permitir esse fluxo de dados de maneira extensível, o IBMC, se valendo da estrutura de orientação a objetos, disponibiliza uma maneira específica para criar os módulos de dados e os de construção. A seguir seguem as explicações sobre esses módulos.

4.2.3 Módulo de Construção.

Os módulos de construções são responsáveis por elaborarem os arquivos de entrada para os códigos de Monte Carlo. Esses módulos são específicos à um determinado código de Monte Carlo, por exemplo o *MCNP*, pois cada código possui um estrutura particular para o seu arquivo de entrada.

Na Figura 4.3 são apresentadas as classes referentes ao módulo de construção. Cada módulo de construção deve ser subclasse de *AbstractInputMonteCarloCode* a qual possui características comuns à todos os módulos. Neste diagrama somente estão contidos os principais métodos da classe *AbstractInputMonteCarloCode*.

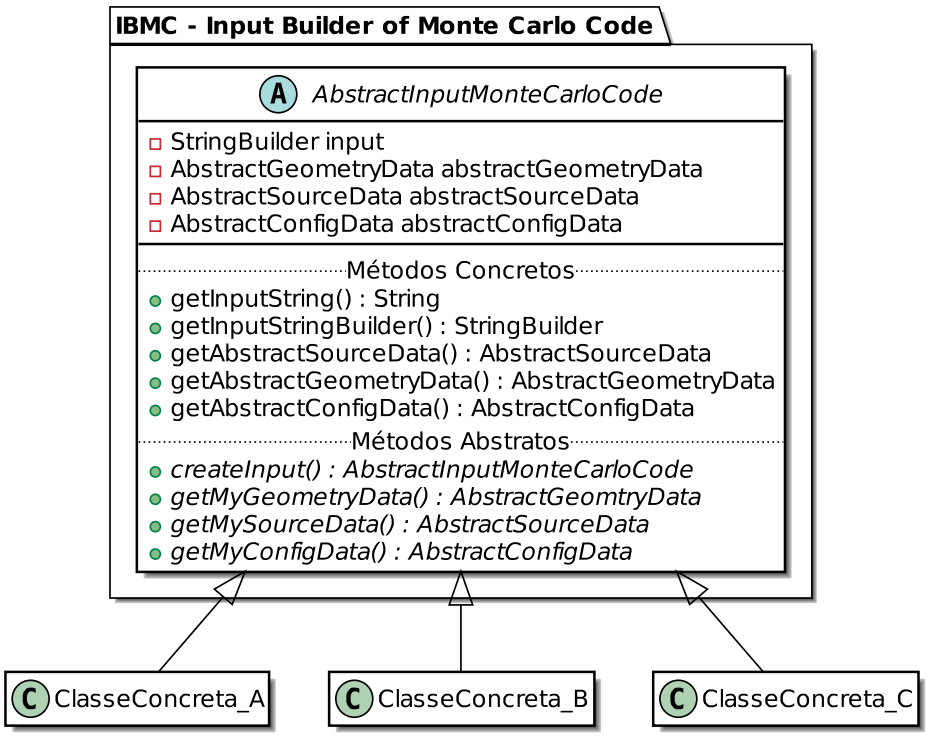

Figura 4.3: Diagrama de classes correspondente aos módulos de construção.

Ela apresenta cinco métodos concretos e quatro abstratos necessários para a elaboração das classes dos módulos de construção. Vide a Tabela 4.1 para explicações mais detalhadas desses métodos.

Os métodos concretos são de uso gerais, sendo úteis a todas as subclasses, já os métodos abstratos, representados em itálico na Tabela 4.1, são específicos à cada subclasse. As diferenças entre os arquivos de entrada, construídos pelos diferentes módulos de construção, são geradas pela implementação desses métodos abstratos.

Tabela 4.1 - Descrição dos cinco métodos concretos e quatro abstratos da classe *AbstractInputMonteCarloCode*.

Métodos	Descrições
getInputString	Fornece um conjunto de caracteres referente ao arquivo de entrada. Útil para gravar em um arquivo o que foi gerado pelo módulo de construção.
getInputStringBuilder	Fornece um objeto de *StringBuilder* referente ao arquivo de entrada. Útil na elaboração do arquivo de entrada pelo módulo de construção.

Continuação da Tabela 4.1

Métodos	Descrições
getAbstratGeometryData	Fornece a classe abstrata de *AbstractGeometryData*. Essa classe representa o módulo de dados referente à geometria da simulação. Será utilizada na implementação do método getMyGeometryData pelas subclasses concretas de *AbstractInputMonteCarloCode*.
getAbstratSourceData	Fornece a classe abstrata de *AbstractSourceData*. Essa classe representa o módulo de dados referente à fonte de radiação. Será utilizada na implementação do método getMySourceData pelas subclasses concretas de *AbstractInputMonteCarloCode*.
getAbstratConfigData	Fornece a classe abstrata de *AbstractConfigData*. Essa classe representa o módulo de dados referente à configurações da simulação. Será utilizada na implementação do método getMyConfigData pelas subclasses concretas de *AbstractInputMonteCarloCode*.
createInput	Método utilizado para criar o arquivo de entrada dos códigos de Monte Carlo.
getMyGeometryData	Método utilizado para especificar qual é a classe referente à geometria que o módulo de construção é capaz de processar.
getMySourceData	Método utilizado para especificar qual é a classe referente à fonte de radiação que o módulo de construção é capaz de processar.
getMyConfigData	Método utilizado para especificar qual é a classe referente à configuração da simulação que o módulo de construção é capaz de processar.

Atualmente, o IBMC possui um único módulo de construção que é destinado ao código *MCNP*. Na Figura 4.4 é mostrado o diagrama de classes para a criação de módulos de construção para o *MCNP*.

Para criar um módulo de construção para o *MCNP* é necessário criar uma subclasse de *InputMCNP*. Ela contêm 6 métodos abstratos que auxiliam na construção do arquivo do *INP* e um método concreto que permite estruturar esse arquivo.

As classes *InputMCNP_Concreta_A*, *B* e *C*, representam diferentes maneiras de se construir o arquivo *INP*. Dessa forma, elas poderiam representar a construção do arquivo *INP* para os diferentes tipos de tratamentos.

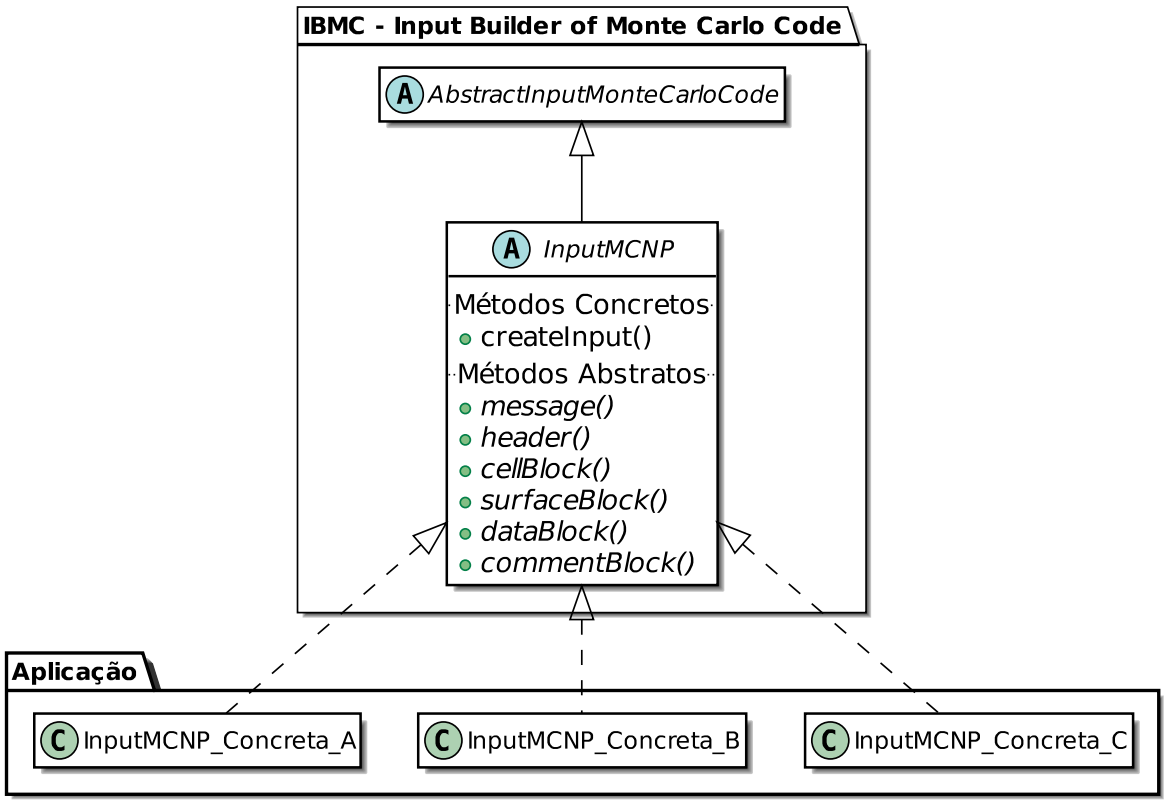

Figura 4.4: Diagrama de Classes para os módulos de construção referente ao *MCNP*.

Na Tabela 4.2 são apresentadas as informações dos principais métodos contidos na classe *InputMCNP* presente na Figura 4.4

As subclasses concretas da classe *InputMCNP* devem necessariamente implementar os métodos abstratos especificados na classe *InputMCNP*. Esses métodos abstratos estão estritamente relacionados com cada parte do arquivo *INP* e o método concreto, *createInput*, processa as informações cadastradas nos métodos abstratos a fim de formar o arquivo *INP*.

Tabela 4.2 - Descrição dos principais métodos da classe *InputMCNP* presente na Figura 4.4.

Métodos	Descrições
createInput	Através dos métodos abstratos, representados nesta tabela em itálico, cria-se a estrutura do arquivo *INP* do *MCNP*.
message	Método utilizado para especificar o que será inserido no bloco de mensagem do arquivo *INP* do *MCNP*.
header	Método utilizado para especificar o que será inserido no cabeçalho do arquivo *INP* do *MCNP*.
cellBlock	Método utilizado para especificar o que será inserido no bloco das células do arquivo *INP* do *MCNP*.

Continuação da Tabela 4.2

Métodos	Descrições
surfaceBlock	Método utilizado para especificar o que será inserido no bloco das superfícies do arquivo *INP* do *MCNP*.
dataBlock	Método utilizado para especificar o que será inserido no bloco de dados do arquivo *INP* do *MCNP*.
commentBlock	Método utilizado para especificar o que será inserido no bloco dos comentários do arquivo *INP* do *MCNP*.

4.2.4 *Módulo de Dados*

A seguir serão descritos quais os módulos de dados e como eles devem ser especificados. O IBMC dispõem de três classes abstratas com o propósito de armazenar os dados de Geometria, Fonte de Radiação e Configurações dos códigos de Monte Carlo, provenientes dos *softwares* terceiros. Assim, para se criar os Módulos de Dados, dentro de uma aplicação particular, é necessário se ater a essas três classes apresentadas. A descrição dessas classes é especificada na Tabela 4.3.

Tabela 4.3 - Representação das descrições das classes referentes aos Módulo de Dados.

Classes	Descrições
AbstractGeometryData	Classe abstrata que tem a responsabilidade de armazenar informações referentes à geometria da simulação.
AbstractSourceData	Classe abstrata que tem a responsabilidade de armazenar informações referentes à fonte de radiação da simulação.
AbstractConfigData	Classe abstrata que tem a responsabilidade de armazenar informações referentes às configurações específicas à simulação. Como por exemplo, materiais, composição, numero de historias, opções de calculo e etc.

Atualmente, as três classes abstratas, referente aos módulos de dados, não possuem atributos e nem métodos. Elas servem apenas como guia para que o IBMC diferencie qual o tipo de dado cada uma representa. A inclusão tanto de atributos quanto métodos poderão ocorrer futuramente caso haja necessidade.

4.2.5 *Vínculo entre os Módulos de Dados e de Construção*

Um fator muito importante do IBMC é que o Módulo de Construção especifica quais os Módulos de Dados que eles são capazes de interpretar. Isto é, quais classes correspondentes aos Módulos de Dados eles são capazes de processar.

Os vínculos são realizados através dos métodos ***getMyGeometryData***, ***getMySourceData*** e ***getMyConfigData*** que pertencem às subclasses de *InputMCNP*. Na Figura 4.5 é apresentado um exemplo de como deve ser a implementação de cada um desses métodos, tendo atenção nas classes dos Módulos de Dados que são retornados por eles. Através desses métodos as subclasses de *InputMCNP* adquirem os dados para elaborar o arquivo *INP*.

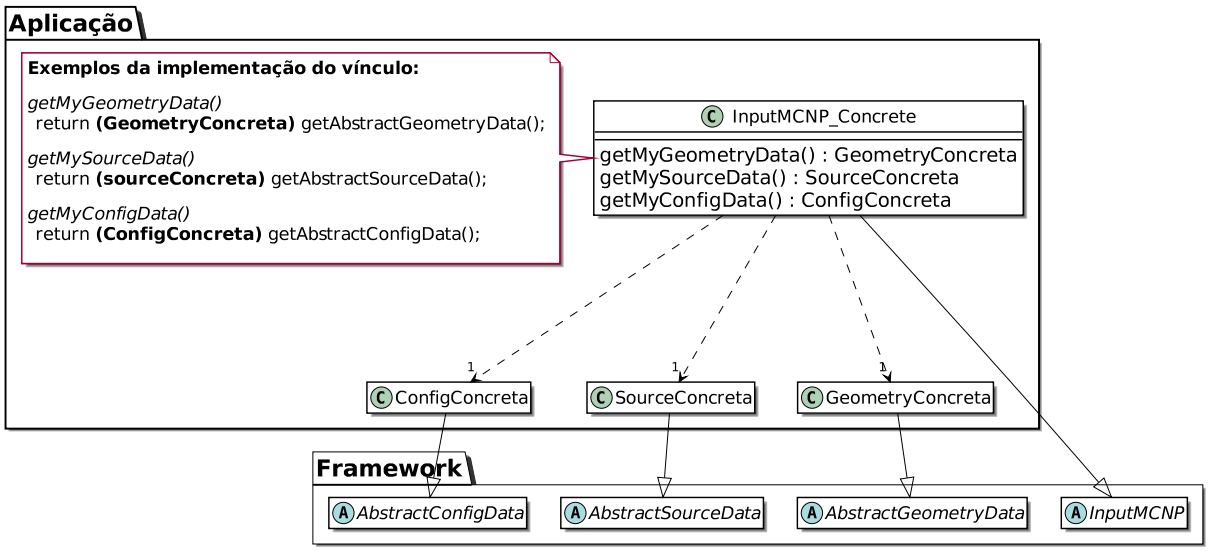

Figura 4.5: Diagrama de Classes que representa o vínculo entre os Módulos de Dados e Construção.

4.2.6 *Exemplo da execução do IBMC para criação do arquivo de entrada do MCNP.*

Outra classe muito importante contida no IBMC é a *SimulationFileManager*. Essa classe é apresentada na Figura 4.6 e seus métodos são detalhados na Tabela 4.4.

Ela fornece um interface única para integrar os Módulos de Dados com o de Construção a fim de gerar o arquivo de entrada para os códigos de Monte Carlo.

Na Tabela 4.4 são mostrados cinco métodos sendo que um deles especifica qual será o Módulo de Construção, três especificam quais serão os Módulos de Dados, tendo atenção que estes devem ser compatíveis com o Módulo de Construção especificado e por fim um

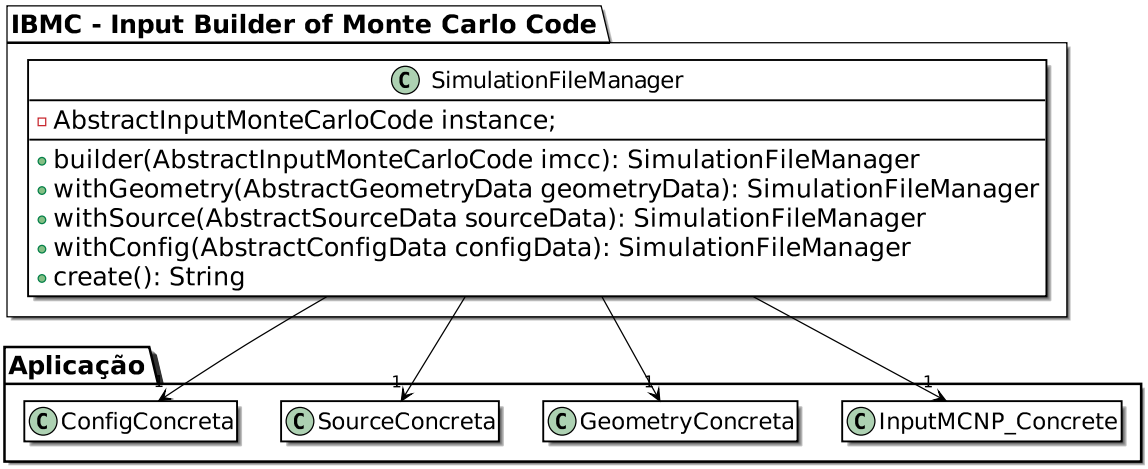

Figura 4.6: Diagrama da classe de *SimulationFileManager*.

método para criar o arquivo e retorná-lo ao cliente na forma de uma *String*.

Tabela 4.4 - Descrição dos métodos referente à classe *SimulationFileManager*.

Métodos	Descrições
buider	Recebe uma instância de algum Módulo de Construção.
withGeometry	Recebe uma instância de algum Módulo de Dados referente à Geometria.
withSource	Recebe uma instância de algum Módulo de Dados referente à Fonte de Radiação.
withConfig	Recebe uma instância de algum Módulo de dados referente à Configuração da Simulação.
create	Inicia a construção do arquivo de entrada e o retorna na forma de uma *String*.

Na Figura 4.7 é apresentado um diagrama de sequência que mostra uma possível sequência de requisições a fim de construir o arquivo *INP* do *MCNP* através do IBMC.

As classes *InputMCNP_Concrete*, *GeometryConcreta*, *SourceConcreta* e *ConfigConcreta* pertencem à aplicação do cliente. Já a classe *SimulationFileManager* pertence ao IBMC. Primeiramente são criados e devidamente preenchidos os objetos dos Módulo de Dados e Construção. Depois disso, é necessário criar o objeto de *SimulationFileManager* que possui métodos, vide Tabela 4.4, para armazenar esses objetos criados anteriormente. Por fim, é feita a requisição da criação do arquivo de entrada mediante o método *create()* o qual retorna o arquivo de entrada na forma de uma *String* que pode ser salva em um arquivo.

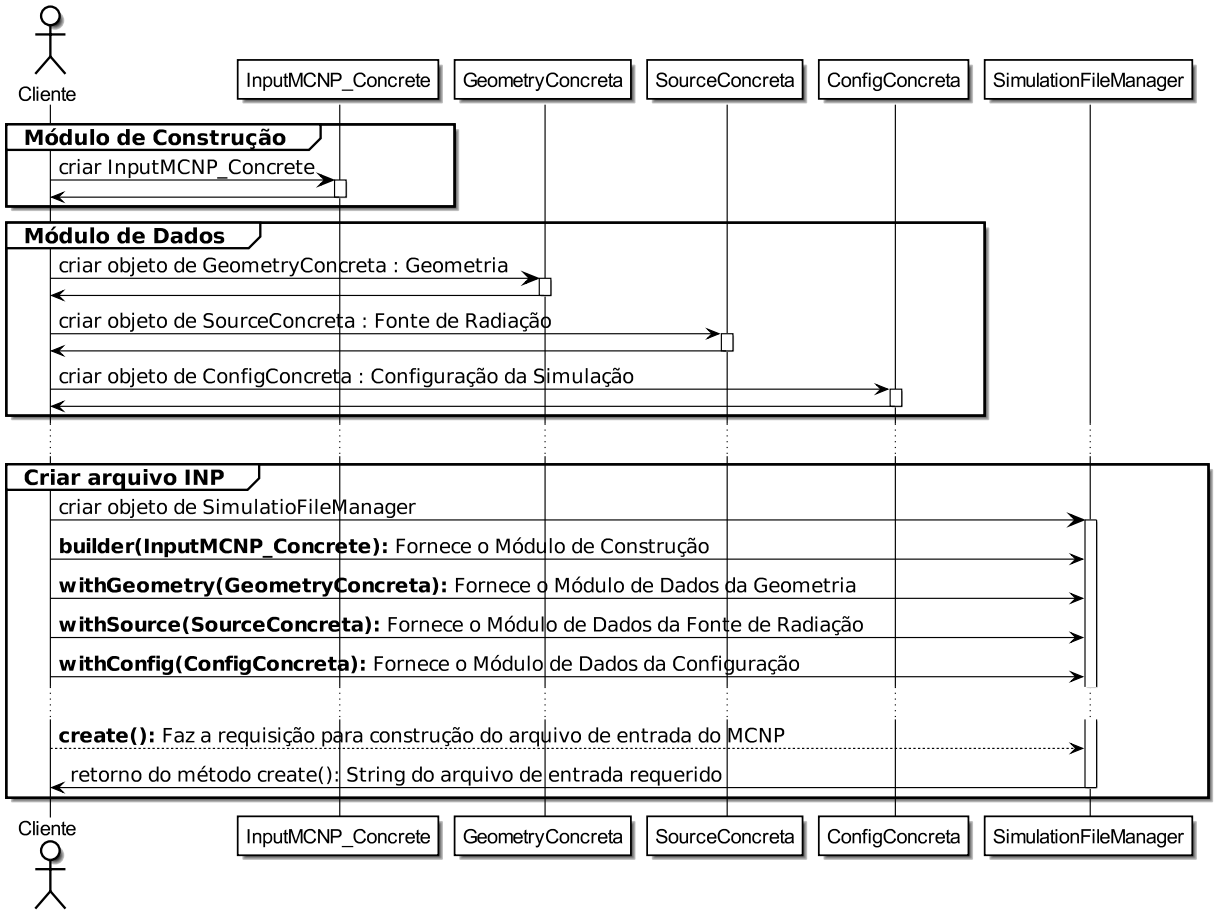

Figura 4.7: Diagrama de sequência para a criação de um arquivo *INP* do *MCNP* através do IBMC.

Capítulo 5

Resultados e Discussões

A seguir serão apresentados os resultados juntamente com suas respectivas discussões.

- Seção 5.1: Exemplos de utilização do sistema desenvolvido com comparações de seus resultados com sistemas já validados.
- Seção 5.2: Casos de uso com o IBMC.

5.1 Exemplos de utilização do IBMC

Foram escolhidos quatro casos correspondentes aos sistemas de planejamento dosimétrico para mostrar o uso do IBMC . Assim, em todos os casos foram executados dois arquivos de entrada. Um deles provenientes de *softwares* já consolidados e o outro proveniente do IBMC.

Há dois fatores importantes a serem observados nos resultados. O primeiro é que um único sistema, desenvolvido através do IBMC, poderia abordar diferentes tipos de tratamentos. O segundo é que uma funcionalidade desenvolvida para um sistema pode ser utilizada em outro.

Os códigos para construção de cada sistema estão no repositório do *Bitbucket*[1] e podem ser acessados livremente através do site.

5.1.1 Medicina Nuclear

Este exemplo retrata um caso fictício de uma simulação de Medicina Nuclear. Foram feitas duas simulações, de um mesmo caso, uma com o *software AMIGO* e a outra com o

[1] <https://bitbucket.org/fmassicano/examples-of-ibmc>

IBMC[2].

A seguir encontram-se as descrições do caso.

- **Geometria**: Dez matrizes de 25 X 25 *voxels* que representam um objeto simulador. Cada valor dessas matrizes correspondem a um material em específico. Ao todo são seis materiais cujas especificações podem ser encontradas na Tabela 5.1. Essas matrizes representam as imagens de Tomografia Computadorizada (CT). Na Figura 5.1 há a representação dessa geometria. Há também outro grupo de dez matrizes de 25 X 25 *voxels* que contêm os valores das densidades para cada *voxels*.

- **Fonte de Radiação**: Dez matrizes de 25 X 25 *voxels* que representam a distribuição da fonte de radiação. Cada valor dessas matrizes representam o número de contagens que é referente ao detector de radiação. A fonte foi especificada para ficar na mesma posição dos materiais três e cinco. Foi considerada uma fonte de Ítrio-90 (^{90}Y), emissor beta puro, e seu espectro foi retirado do *NIST* (*National Institute of Standards and Technology*) (84). Essas matrizes representam as imagens de *SPECT* (*Single Photon Emission Computed Tomography*) ou PET (*Positron Emission Tomography*).

- ***Tally***: Calculado pelo *tally *f8*, o qual fornece a energia depositada em *MeV*. Foram somente consideradas a terceira, quarta e quinta *matrizes*.

[2] <https://bitbucket.org/fmassicano/examples-of-ibmc/src/8996b8bcd7fd7287db4b377aa7c874043c9b745b/src/main/java/examples/nuclear_medicine/first_case/?at=master>

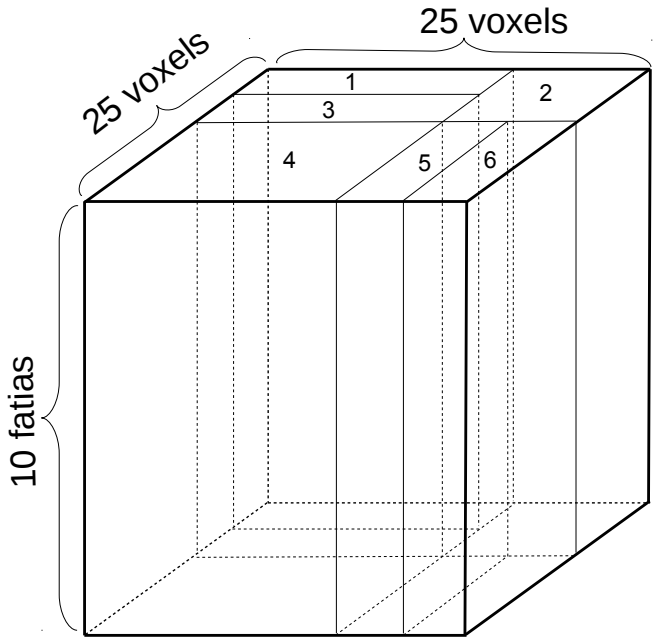

Figura 5.1: Esquema da geometria do problema de medicina nuclear.

Tabela 5.1 - Composição dos materiais referente à simulação de Medicina Nuclear

		Pesos Percentuais								
Material	ρ (g/cm^3)	H	C	N	O	Na	P	S	Cl	Ca
1	0,296	10,3	10,5	3,10	74,90	0.20	0,20	0,30	0,30	0,20
2	0,880	12,1	89,6	1,10	0,80	0,20	-	-	-	0,10
3	0,960	11,3	56,4	0,90	31,00	0,10	0,10	0,10	0,10	-
4	0,980	11,0	45,7	1,60	41,30	-	0,10	0,20	0,20	0,10
5	1,010	10,8	35,5	2,20	51,00	-	0,10	0,20	0,20	0,10
6	1,030	10,6	28,9	2,60	57,30	-	0,10	0,20	0,20	0,10

A seguir serão descritos os passos para gerar o arquivo do *MCNP* mediante o IBMC. O procedimento adotado aqui pode ser utilizado para um caso de simulação de um paciente real.

Criar Módulo de Dados referente à Geometria: O primeiro passo foi elaborar uma classe correspondente ao módulo de dados da geometria. A seguir encontra-se as descrições desse procedimento.

Primeiramente, é suposto haver um *software* terceiro que forneça todas as informações da geometria a fim de permitir a simulação no *MCNP*.

Suponha que esse *software* terceiro, forneça a geometria através de três arquivos. Um, denominado ***phantom***, contendo matrizes que apresentam os identificadores do tecido o outro, denominado ***density***, contendo matrizes que apresentam a densidade dos tecidos e

por fim um arquivo, denominado ***icctRange.xml*** que contêm as composições dos materiais. Esse *software* provavelmente utilizará imagens de Tomografia Computadorizada para construir esses arquivos. Dentro destes arquivos as matrizes se encontram uma abaixo da outra.

É valido mencionar que o Centro de Engenharia Nuclear (CEN) do Instituto de Pesquisas Energéticas e Nucleares (IPEN) possui alguns softwares, por exemplo, *ICCT* e *AMIGOBrachy* que já poderiam ser utilizados em conjunto com o IBMC.

Para interpretar esses arquivos foi elaborada a classe *GeometryDataMCNP*, referente ao módulo de dados. Ela é a responsável por armazenar as informações provenientes do *software* terceiro.

Essa classe futuramente será utilizada pelo módulo de construção para assim formar o arquivo do *MCNP*.

A seguir, na Figura 5.2, há o diagrama de classes de *GeometryDataMCNP*. A única especificação para esta classe é ela ser subclasse de *AbstractGeometryData*.

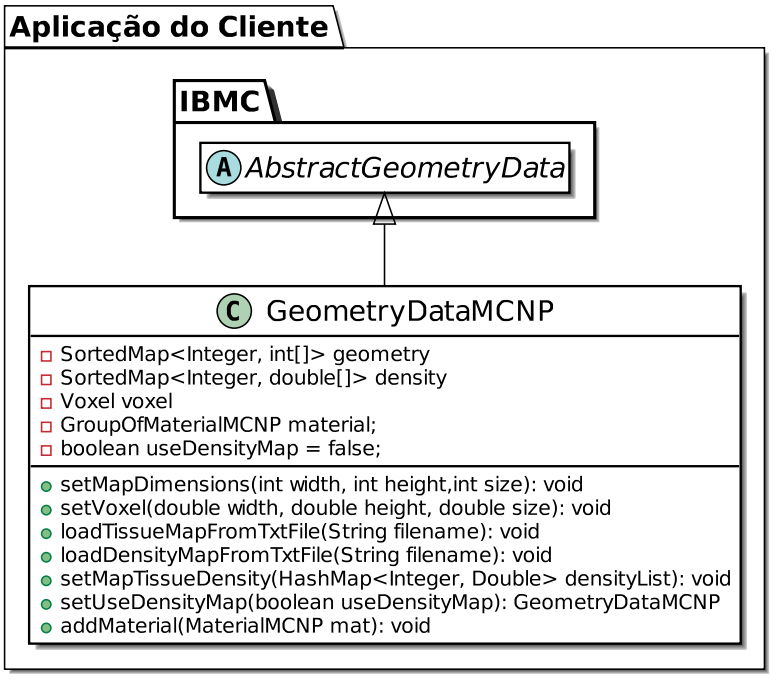

Figura 5.2: Diagrama de classes de *GeometryDataMCNP*.

A descrição de cada método da classe *GeometryDataMCNP* está contida na Tabela 5.2.

Tabela 5.2 - Descrição dos métodos da classe *GeometryDataMCNP*.

Métodos	Descrições
setMapDimensions	Este método recebe três parâmetros que especificam as dimensões da matriz de geometria.
setVoxel	Este método é utilizado para especificar as dimensões do *voxel*.
loadTissueMapFromTxtFile	Este método recebe o caminho do arquivo que contêm as matrizes de geometria e carrega essas matrizes da geometria em um objeto de *SortedMap*. Esse objeto é um mapa de chave valor em que a chave é o número da matriz e o valor é um *array* contendo a matriz da geometria vetorizada.
loadDensityMapFromTxtFile	Este método recebe o caminho do arquivo que contêm as matrizes de densidade e carrega as matrizes da densidade em um objeto de *SortedMap*. Esse objeto é um mapa de chave valor em que a chave é o número da matriz e o valor é um *array* contendo a matriz de densidade vetorizada.
setUseDensityMap	Este método recebe um valor booleano. Caso seja "*true*" o mapa de densidade será levado em consideração, caso seja "*false*" deve-se especificar através do método *setMapTissueDensity* um mapa de chave valor sendo que a chave é o número do tecido e o valor a densidade correspondente desse tecido. Ele possui valor padrão como "*true*".
setMapTissueDensity	Este método recebe um objeto da classe *HashMap*, que representa uma mapa de chave e valor. A chave é o número do tecido e o valor a densidade desse tecido. Este método é utilizado quando não se deseja utilizar o mapa de densidade.
addMaterial	Este método recebe um objeto da classe *MaterialMCNP*, descrito em C.1.5, da biblioteca *helperMCNP*. Através desse método é possível cadastrar os materiais da simulação.

A seguir, na Figura 5.3, há o diagrama de sequência que exemplifica como a classe *geometryDataMCNP* é utilizada para extrair as informações dos arquivos que contêm os dados da geometria.

Primeiramente é criado o objeto "*gd*" da classe *GeometryDataMCNP*. Depois é utilizado o método *setMapDimensions* para definir as dimensões das matrizes contidas no arquivo *phantom* e *density*. Estas duas matrizes tridimensionais devem possuir as mesmas dimensões.

O próximo passo é definir qual o tamanho do *voxel*. Em seguida, são utilizados os métodos *loadTissueMapFromTxtFile* e *loadDensityMapFromTextFile* a fim de carregar as matrizes de tecido e densidade, respectivamente.

O último passo é ler o arquivo ***icctRange.xml*** e adicionar a lista de materiais através do método *addMaterial*.

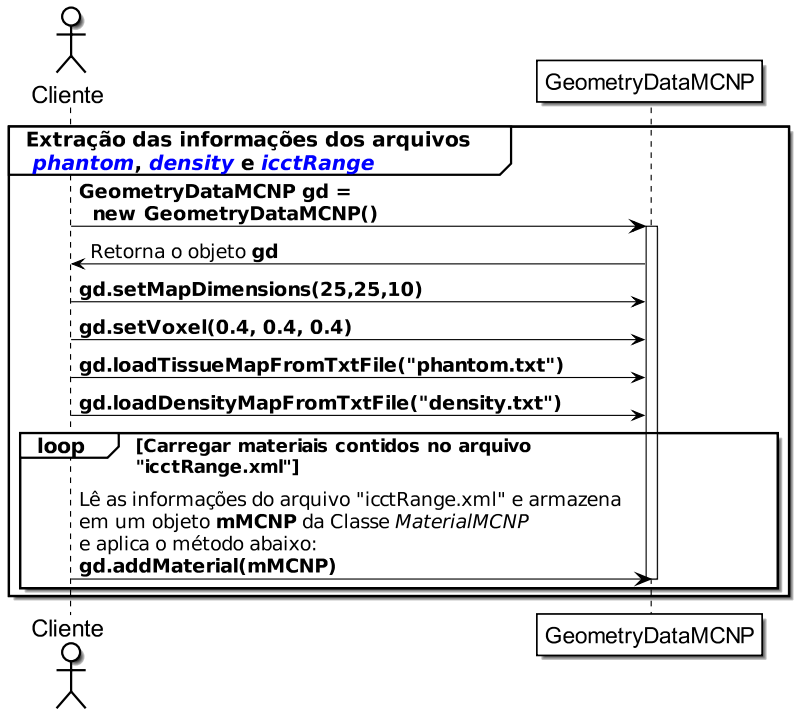

Figura 5.3: Diagrama de sequência que exemplifica o uso da classe *GeometryDataMCNP* para extrair as informações provenientes do *software* terceiro.

Criar Módulo de Dados referente à Fonte de Radiação: O próximo passo foi a elaboração da classe *SourceDataMCNP* correspondente ao módulo de dados da fonte de radiação. A seguir encontra-se as descrições desse procedimento.

Da mesma forma que no Módulo de Dados da geometria, é suposto que haja um *software* terceiro responsável por adquirir todos os dados da fonte de radiação a fim de permitir a simulação no *MCNP*.

Suponha que este *software* forneça a fonte de radiação através de um arquivo, denominado ***source***. Dentro deste arquivo há matrizes contendo os valores da fonte de radiação, sendo dispostas uma abaixo da outra, dentro do arquivo ***source***. Os dados do espectro do ^{90}Y são fornecidos através de um arquivo, denominado ***spectrum***, contendo o par de valores energia e frequência.

Para armazenar os dados provenientes desse *software* terceiro foi elaborada a classe *SourceDataMCNP* que será futuramente utilizada pelo módulo de construção para assim formar o arquivo do *MCNP*.

A seguir, na Figura 5.4, há o diagrama de classes de *SourceDataMCNP*. A única especificação para esta classe é ela ser subclasse de *AbstractSourceData*.

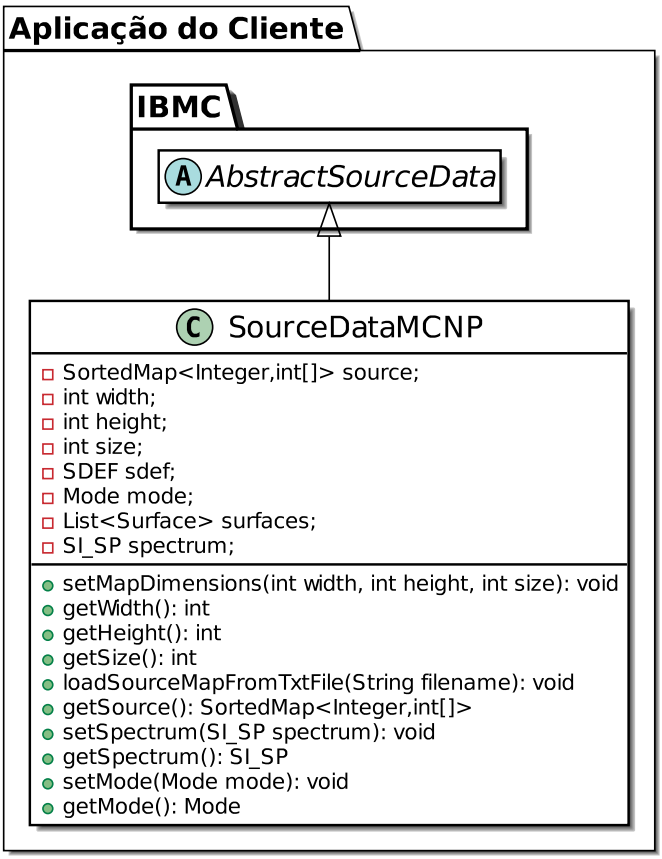

Figura 5.4: Diagrama de classes de *SourceDataMCNP*.

A descrição de cada método da classe *SourceDataMCNP* está contida na Tabela 5.3.

Tabela 5.3 - Descrição dos métodos da classe *SourceDataMCNP*.

Métodos	Descrições
setMapDimensions	Este método recebe três parâmetros que especificam as dimensões da matriz de geometria.
getWidth	Retorna o valor da dimensão referente à *width*.
getHeight	Retorna o valor da dimensão referente à *height*.
getSize	Retorna o valor da dimensão referente à *size*.

Continuação da Tabela 5.3

Métodos	Descrições
loadTissueMapFromTxtFile	Este método recebe o caminho do arquivo que contêm as matrizes de fonte de radiação e carrega essas matrizes em um objeto de *SortedMap*. Esse objeto é um mapa de chave valor em que a chave é o número da matriz e o valor é um *array* contendo a matriz vetorizada da fonte.
getSource	Retorna o objeto de SortedMap que foi armazenado pelo método *loadTissueMapFromTxtFile*.
setSpectrum	Este método recebe um objeto da Classe *SI_SP* que definirá o espectro da fonte radioativa.
getSpectrum	Retorna o espectro mediante a classe *SI_SP*.
setMode	Este método recebe um objeto da Classe *Mode* que definirá quais partículas serão simuladas.
getMode	Retorna o cartão *Mode* através da classe *Mode*.

Na Figura 5.5 contêm um diagrama de sequência que exemplifica como a classe *SourceDataMCNP* poderá ser utilizada para extrair as informações contidas no arquivo de texto *source*.

Primeiramente tem-se a criação da classe *SourceDataMCNP*. Depois são configuradas as dimensões da matrizes da fonte com o método *setMapDimensions* passando os valores 25, 25 e 10. Após a definição da geometria a matriz contida no arquivo *source* é carregada através do método *loadSourceMapFromTextFile*.

O espectro de radiação contido no arquivo *spectrum* é interpretado pelo *software* do cliente extraindo o par de informações energia e frequência. Esses valores são carregados no objeto da classe *SI_SP* e depois passados à *SourceDataMCNP* mediante o método *setSpectrum*.

Por fim é feita a especificação do cartão *Mode* por intermédio do método *setMode*. Nesse caso, foi especificado o transporte de fótons e elétrons.

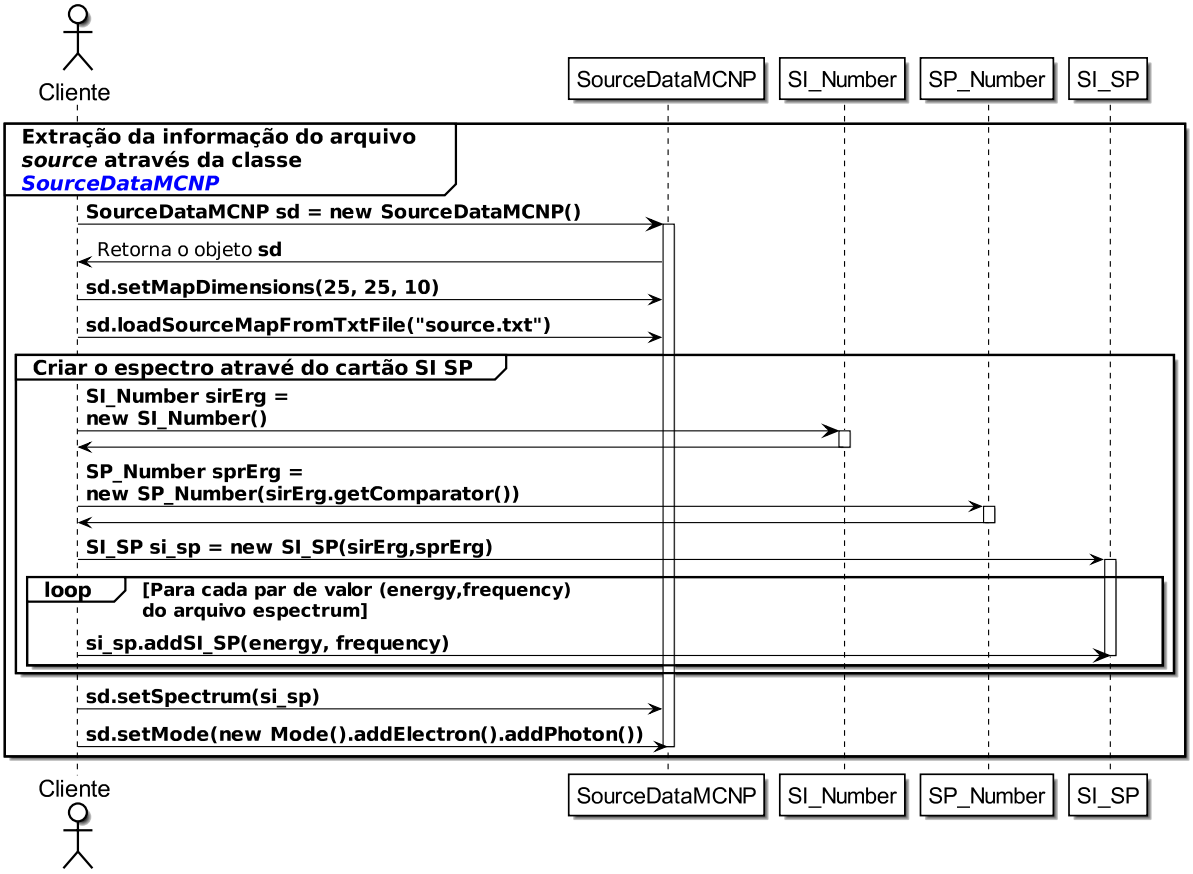

Figura 5.5: Diagrama de sequência que exemplifica o uso da classe *SourceDataMCNP* para extrair as informações do arquivo *source*.

Criar Módulo de Dados referente à Configuração do MCNP: O terceiro passo foi elaborar uma classe correspondente ao módulo de dados da configuração. Foi construída a classe *configDataMCNP* para interpretar as informações da configuração do *MCNP*. A seguir encontra-se as descrições desse procedimento.

Primeiramente, é suposto haver um *software* terceiro que forneça todas as informações de configuração a fim de permitir a simulação no *MCNP*.

Suponha que este *software* forneça as informações de configuração através de dois arquivos, um denominado ***tally***, com as informações do *tally* e o outro, denominado ***config***, com algumas especificações referente a simulação.

No arquivo ***tally*** há uma matriz tridimensional de mesma dimensões que a da geometria com valores zero e um. Os valores zero indicam ausência de *tally* e os valores um a presença. Para esse caso de medicina nuclear o *tally* foi especificado pelo cartão **f8*, o qual fornece a energia deposita em *MeV*. Essa informação está contida na classe responsável por criar o arquivo *INP* do *MCNP*. Foi estabelecido dessa forma pois no caso da Medicina Nuclear

não é habitual haver a utilização de outro cartão para especificar o *Tally*.

No arquivo ***config***, apresentado na Figura 5.7, há informações para os cartões *NPS*, *PRDMP* e *PRINT* do *MCNP*.

Na Figura 5.6 há o diagrama de classes referente ao *Tally*.

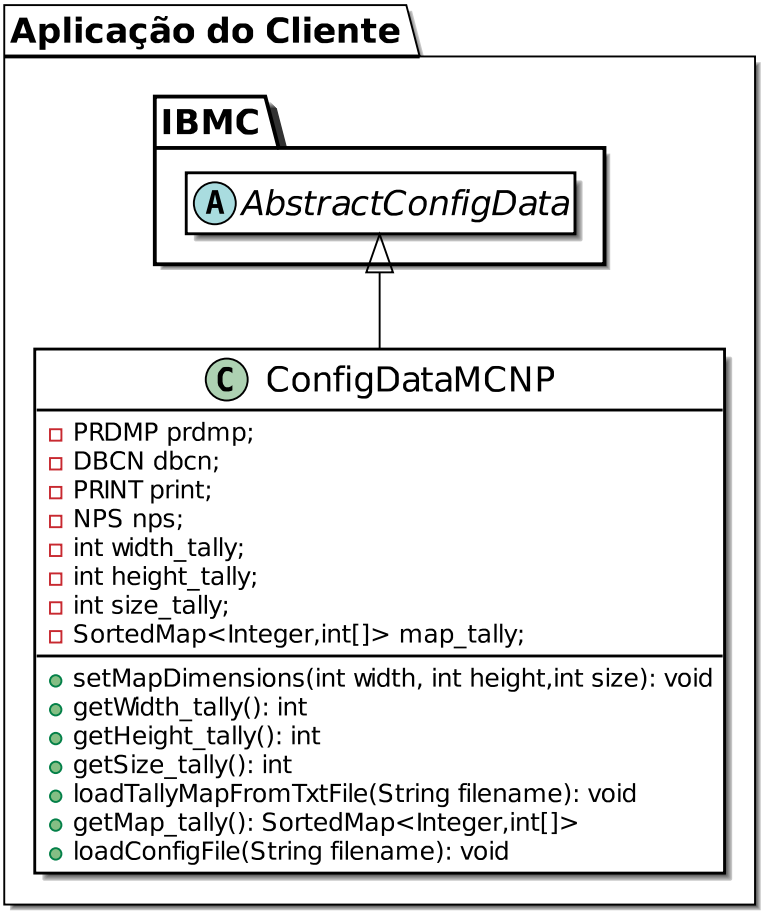

Figura 5.6: Diagrama de classes de *ConfigDataMCNP*.

A seguir na Tabela 5.4 há as descrições dos métodos da classe *ConfigDataMCNP*.

Tabela 5.4 - Descrição dos métodos da classe *ConfigDataMCNP*.

Métodos	Descrições
setMapDimensions	Este método recebe três parâmetros que especificam as dimensões da matriz do *Tally*.
getWidth_tally	Retorna o valor da dimensão referente à *width*.
getHeight_tally	Retorna o valor da dimensão referente à *height*.
getSize_tally	Retorna o valor da dimensão referente à *size*.

Continuação da Tabela 5.4

Métodos	Descrições
loadTallyMapFromTxtFile	Este método recebe o caminho do arquivo que contêm as matrizes do *Tally* e carrega essas matrizes em um objeto de *SortedMap*. Esse objeto é um mapa de chave valor em que a chave é o número da matriz e o valor é um *array* contendo a matriz vetorizada do *Tally*.
getTallyMap	Retorna o objeto *SortedMap* com as matrizes do *Tally*.
loadConfigFile	Este método recebe o caminho do arquivo que contêm as configurações do *MCNP*. Este método somente interpreta valores do cartão *NPS*, *PRDMP* e *PRINT*. Na Figura 5.7 há um exemplo do formato desse arquivo de configuração.

```
nps NPP=1E+6
prdmp NDM=5e+6 MCT=1 NDMP=3
print 30 110
```

Figura 5.7: Esquema da formato do arquivo de configuração que contêm informações para os cartões *NPS* , *PRDMP* e *PRINT*.

O diagrama de sequência mostrado na Figura 5.8 exemplifica o uso da classe *ConfigDataMCNP* a fim de extrair as informações de configuração do *MCNP*.

Primeiramente é criado o objeto ***configData*** da classe *ConfigDataMCNP*. Antes de carregar a matriz do *Tally* é necessário especificar através do método *setMapDimension* as dimensões dessa matriz. Após esse procedimento a matriz contida no arquivo *tally* pode ser carregada mediante o método *loadTallyMapFromTextFile*. Por fim a leitura dos dados contidos no arquivo *config* é feita através do método *loadConfigFile*.

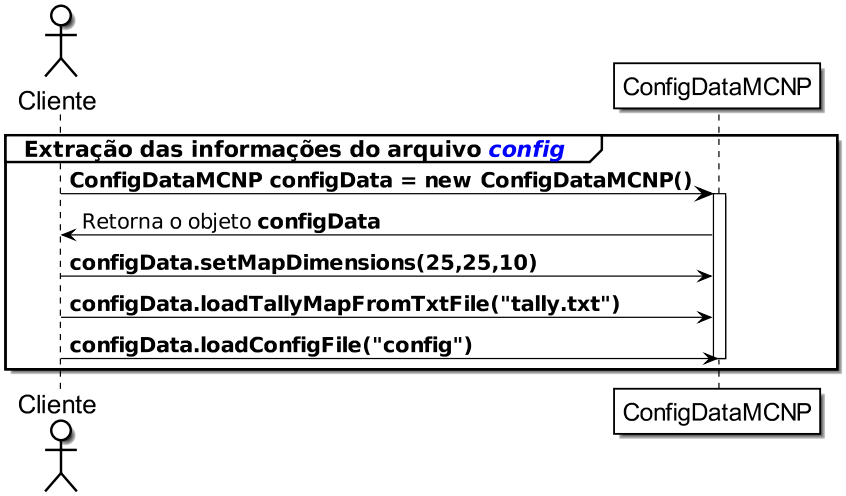

Figura 5.8: Diagrama de sequência que exemplifica o uso da classe *ConfigDataMCNP* para extrair as informações do arquivo *tally* e *config*.

Criar Módulo de Construção: O quarto passo foi elaborar uma classe correspondente ao módulo de construção. Ela foi nomeada de *NuclearMedicineMcnpx*. A seguir foram descritos as implementações de cada um de seus métodos correspondentes à sua subclasse *InputMCNP*.

As explicações de cada método contidos nessa classe estão na Tabela 4.2. O primeiro método abordado foi o *message*. A Figura 5.9 apresenta um diagrama de sequência com a ilustração de sua implementação. Como já mencionado, na seção 4.2.3, sobre o Módulo de construção, vê-se que para adicionar qualquer comando do *MCNP*, como por exemplo, "*message: o=nmm.o r=nmm.r*", é utilizado o método *getInputStringBuilder* da classe *AbstractInputMonteCarloCode*.

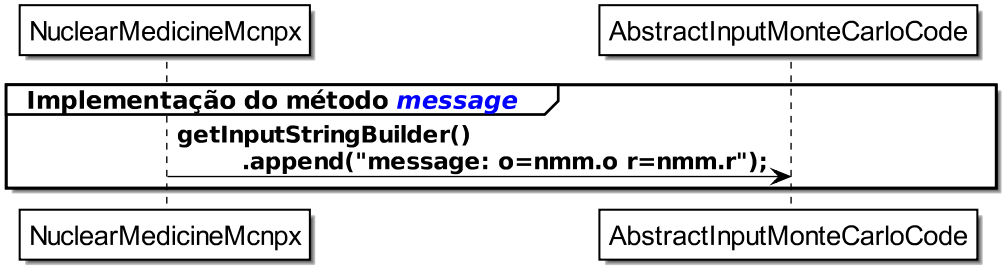

Figura 5.9: Diagrama de sequência da implementação do método *message*.

O segundo método implementado foi o *header*. A Figura 5.10 apresenta seu diagrama de sequência.

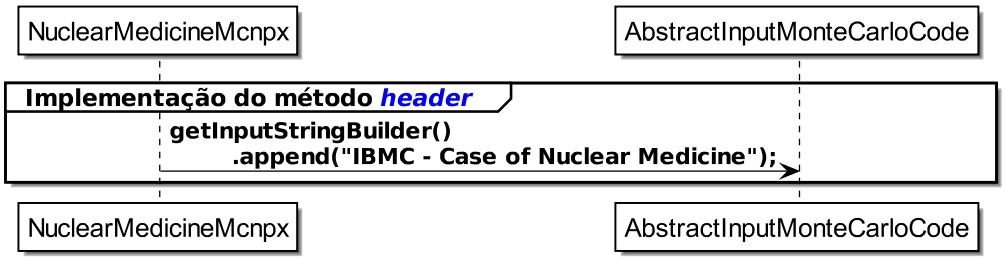

Figura 5.10: Diagrama de sequência da implementação do método *header*.

O terceiro método implementado foi o *cellBlock*. Como já é sabido que este caso utilizará estruturas repetidas e que os demais casos também especificarão as estruturas repetidas, foi elaborada a classe *RepeatedStructureForPhantom*. Esta classe encapsula a criação de estruturas repetidas. Com isso, outros casos podem se beneficiar dessa classe.

Na Figura 5.11 há o diagrama de classes de *RepeatedStructureForPhantom*. Somente foram apresentados alguns de seus métodos.

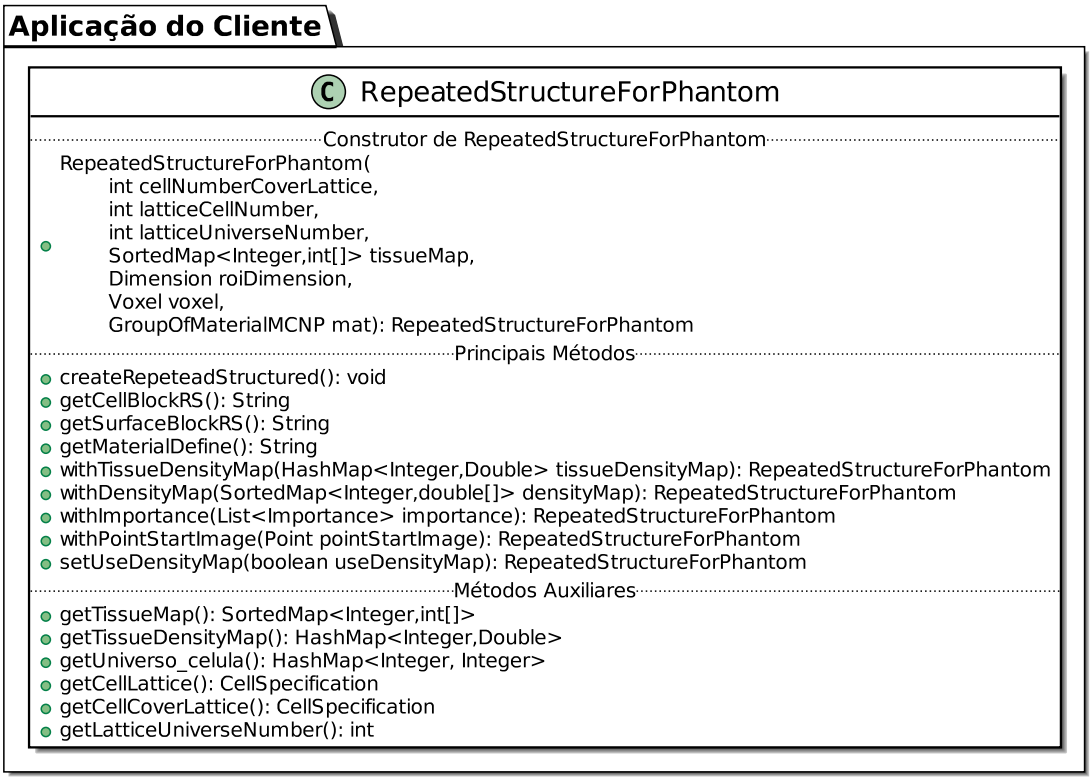

Figura 5.11: Diagrama de classe de *RepeatedStructureForPhantom*.

Na Tabela 5.5 há as descrições dos métodos da classe *RepeatedStructureForPhantom* apresentados no diagrama de classe da Figura 5.11.

Tabela 5.5 - Descrição dos métodos da classe *RepeatedStructureForPhantom*.

Métodos	Descrições
Construtor	O construtor de *RepeatedStructureForPhantom* recebe sete parâmetros. O primeiro é o número da célula que ira cobrir a célula do *Lattice*. O segundo é o número da célula do *Lattice*. O terceiro é o número do universo que se deseja colocar na célula do *Lattice*. O quarto é um objeto de *SortedMap*, que é um mapa de **chave** e **valor**. O valor da **chave** é correspondente ao número da matriz de tecidos e o **valor** é um *array* de inteiros com os valores dos tecidos. O quinto é um objeto da classe *Dimension* que especifica a dimensão da região de interesse que se deseja simular. O sexto é um objeto de *Voxel* que especifica as dimensões do *voxel* que será criado com as estruturas repetidas. O sétimo e último é um objeto de *GroupOfmaterialMCNP* que especifica os materiais contidos no objeto simulador.
createRepeteadStructured	Esse método executa a construção da estrutura repetida a fim de formar o objeto simulador.
getCellBlockRS	Esse método fornece a representação, no formato do *MCNP*, de todas as células envolvidas na construção da estrutura repetida.
getSurfaceBlockRS	Esse método fornece a representação, no formato do *MCNP*, de todas as superfícies envolvidas na construção da estrutura repetida.
getMaterialDefine	Esse método fornece a representação, no formato do *MCNP*, dos materiais envolvidos na construção da estrutura repetida.
withTissueDensityMap	Esse método recebe um objeto de *HashMap* que é um mapa **chave valor**. Os valores das **chaves** representam o tecido enquanto os dos **valores** representam a densidade.
withDensityMap	Esse método recebe um objeto de *SortedMap*. A **chave** representa o número da matriz de densidade e o **valor** representa a respectiva matriz vetorizada.
withImportance	Esse método recebe uma Lista objeto da classe *Importance* a fim de indicar quais as importâncias em todas as células da simulação.
setUseDensityMap	Esse método recebe um valor *booleano* que quando especificado como *true* indica se a classe *RepeatedStructureForPhantom* irá utilizar o mapa de densidade, declarado mediante o método *withDensityMap* ou quando false irá utilizar valores fixos à cada tecido, declarado pelo método *withTissueDensityMap*.

Continuação da Tabela 5.5

Métodos	Descrições
withPointStartImage	Recebe um objeto de *Point* que indica a localização de um ponto nas coordenadas cartesianas. Considerando primeiramente, a matriz de tecidos como sendo uma imagem de *voxels*, tem-se que esse objeto *Point* é a localização de um *voxel* na imagem. Especificamente esse ponto é o ponto inicial situado no canto superior esquerdo da imagem. Através desse ponto e da dimensão da região de interesse é possível especificar uma determinada região a ser simulada.
getTissueMap	Fornece o mapa de tecido na forma de um objeto de *SortedMap*.
getUniverso_celula	Fornece um objeto de *SortedMap* em que a **chave** é o número de um universo e o **valor** é o número da célula que contêm esse universo.
getCellLattice	Fornece um objeto de *CellSpecification* que corresponde à célula do *Lattice*.
getCellCoverLattice	Fornece um objeto de *CellSpecification* que corresponde à célula que envolve a célula do *Lattice*.
getLatticeUniverseNumber	Fornece o valor do universo da célula do *Lattice*.

A Figura 5.12 apresenta o diagrama de sequência do método *cellBlock*. Esse método utilizou a classe *RepeatedStructureForPhantom*.

Primeiramente é criado o objeto ***rsfphantom*** da classe *RepeatedStructureForPhantom*. Para criar esse objeto são utilizados os métodos da classe *GeometryDataMCNP* a qual é acessada pelo método *getMyGeometryData*. Neste parte fica evidente as responsabilidades do Módulo de Dados, representado pela classe *GeometryDataMCNP*, e do Módulo de Construção, representado por *NuclerMedicineMcnpx*. O Módulo de Dados adquiri as informações dos *softwares* terceiros e é utilizado pelo Módulo de Construção a fim de gerar o arquivo do *MCNP*.

Após a construção do objeto ***rsfphantom*** o método *cellBlock* adiciona o mapa de densidade, outro mapa de densidade porém com valores fixos para cada tecido e as importâncias das partículas. Em *cellBlock* é também especificado o método *setUseDensityMap* com valor proveniente do objeto da classe *GeometryDataMCNP*. Assim o objeto de *GeometryDataMCNP* diz qual mapa de densidade deve ser utilizado.

O penúltimo passo é criar a representação da estrutura repetida pelo método *create-*

RepeteadStructured. Para que esta representação da célula da estrutura repetida apareça no arquivo de entrada do *MCNP* é utilizado o método *getCellBlockRS* juntamente com o método *getInputStringBuilder*.

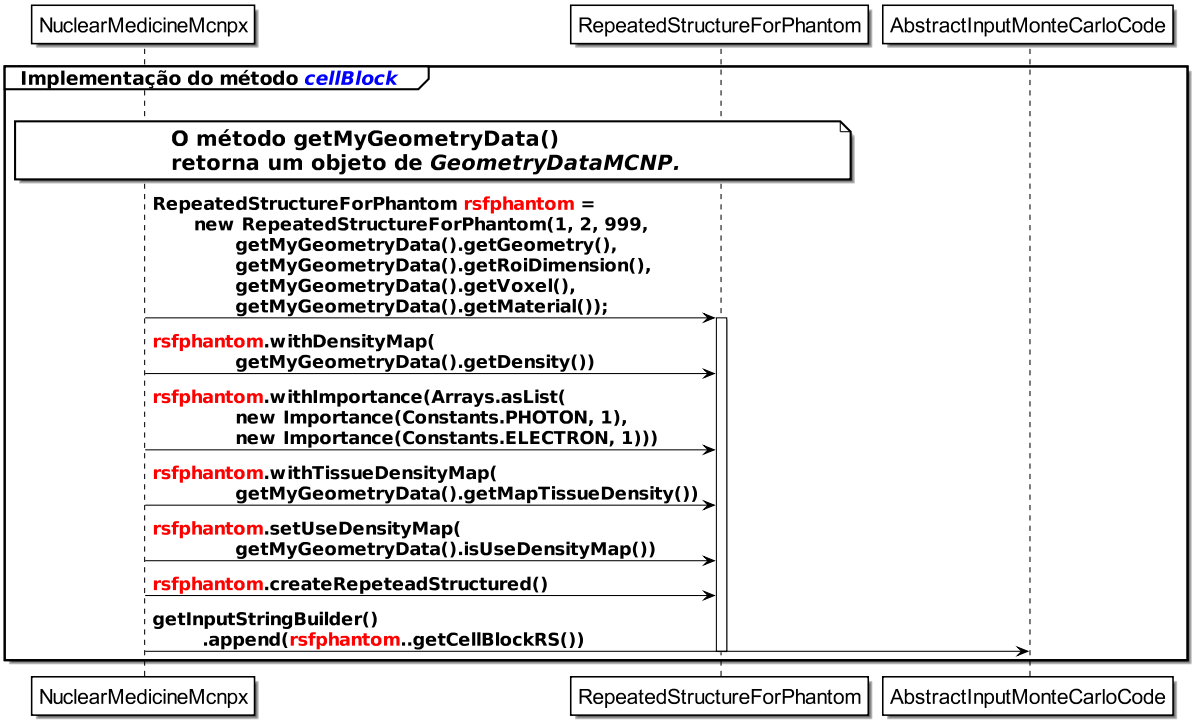

Figura 5.12: Diagrama de sequência da implementação do método *cellBlock*.

O quarto método implementado foi o *surfaceBlock*. Na Figura 5.13 há um diagrama de sequência que mostra sua implementação.

Sua implementação é simples uma vez que o objeto ***rsfphantom*** já foi criado em *cellBlock*.

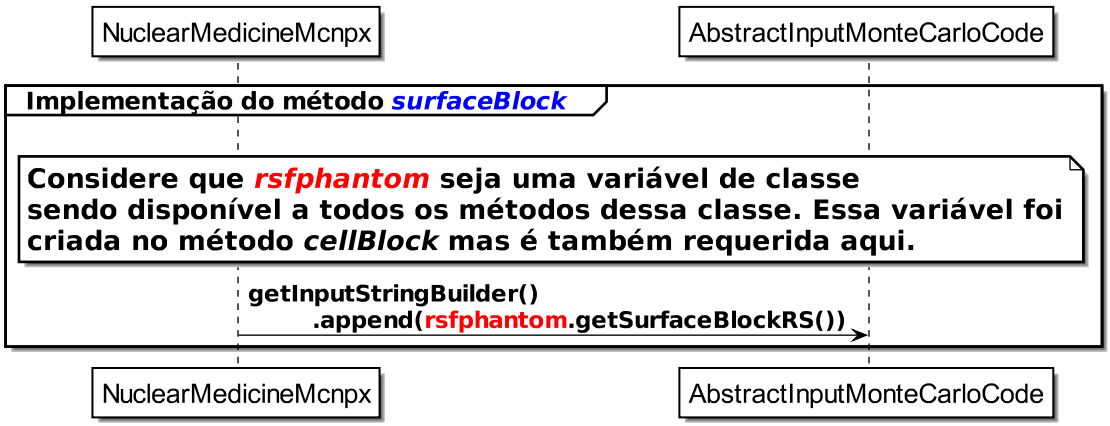

Figura 5.13: Diagrama de sequência da implementação do método *surfaceBlock*.

O quinto método implementado foi o *dataBlock*. Neste método foram implementadas

as informações da fonte de radiação, do *Tally*, dos materiais utilizados na simulação e por fim informações da configuração da simulação.

Para implementar as informações da fonte de radiação foi elaborada a classe *RepeatedStructureForSource* sendo seu diagrama de classe apresentado na Figura 5.14. Somente os métodos mais importantes foram apresentados.

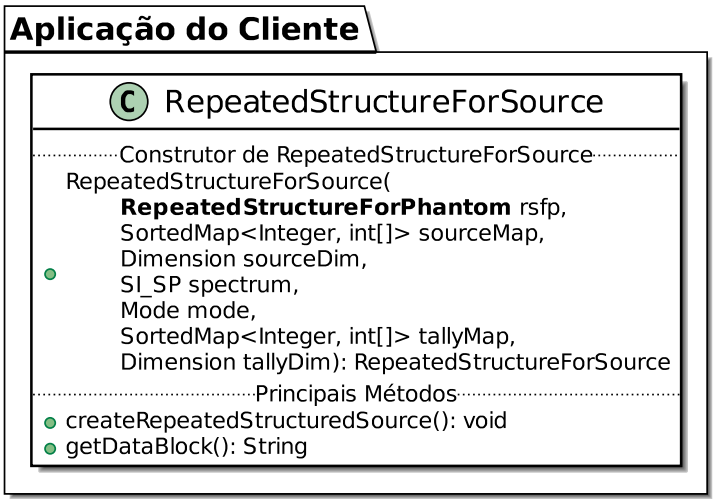

Figura 5.14: Diagrama de classe de *RepeatedStructureForSource*.

Na Tabela 5.6 há as descrições dos métodos apresentados na Figura 5.14.

Tabela 5.6 - Descrição dos métodos da classe *RepeatedStructureForSource*.

Métodos	Descrições
Construtor	O construtor de *RepeatedStructureForSource* recebe sete parâmetros. O primeiro é o objeto da classe *RepeatedStructureForSource*. O segundo é um objeto de *SortedMap*, que é um mapa de **chave** e **valor**. O valor da **chave** é correspondente ao número da matriz da fonte de radiação e o **valor** a respectiva matriz vetorizada. O terceiro é um objeto da classe *Dimension* que especifica a dimensão da região de interesse que se deseja simular. O quarto é um objeto de *SI_SP* que declara o espectro de radiação. O quinto é um objeto *Mode* para dizer quais partículas serão simuladas. O sexto é um objeto de *SortedMap*, que é um mapa de **chave** e **valor**, onde o valor da **chave** é correspondente ao número da matriz do *Tally* e o **valor** a respectiva matriz vetorizada. O sétimo e último é um objeto de *Dimension* que especifica a dimensão do *Tally*.
createRepeteadStructuredSource	Esse método constrói a fonte de radiação mediante com base nas células da estrutura repetida.

Continuação da Tabela 5.6

Métodos	Descrições
getDataMCNP	Esse método fornece a representação, no formato do *MCNP*, da fonte de radiação e do *Tally*. Não há como especificar outro *Tally* mediante esta classe. Ele está fixo para ser o **F8*.

Com base na classe *RepeatedStructureForSource* e com as informações de *ConfigDataMCNP* e *RepeatedStructureForPhantom* foi elaborado o método *dataBlock*. Na Figura 5.15 há um diagrama de sequência da implementação desse método.

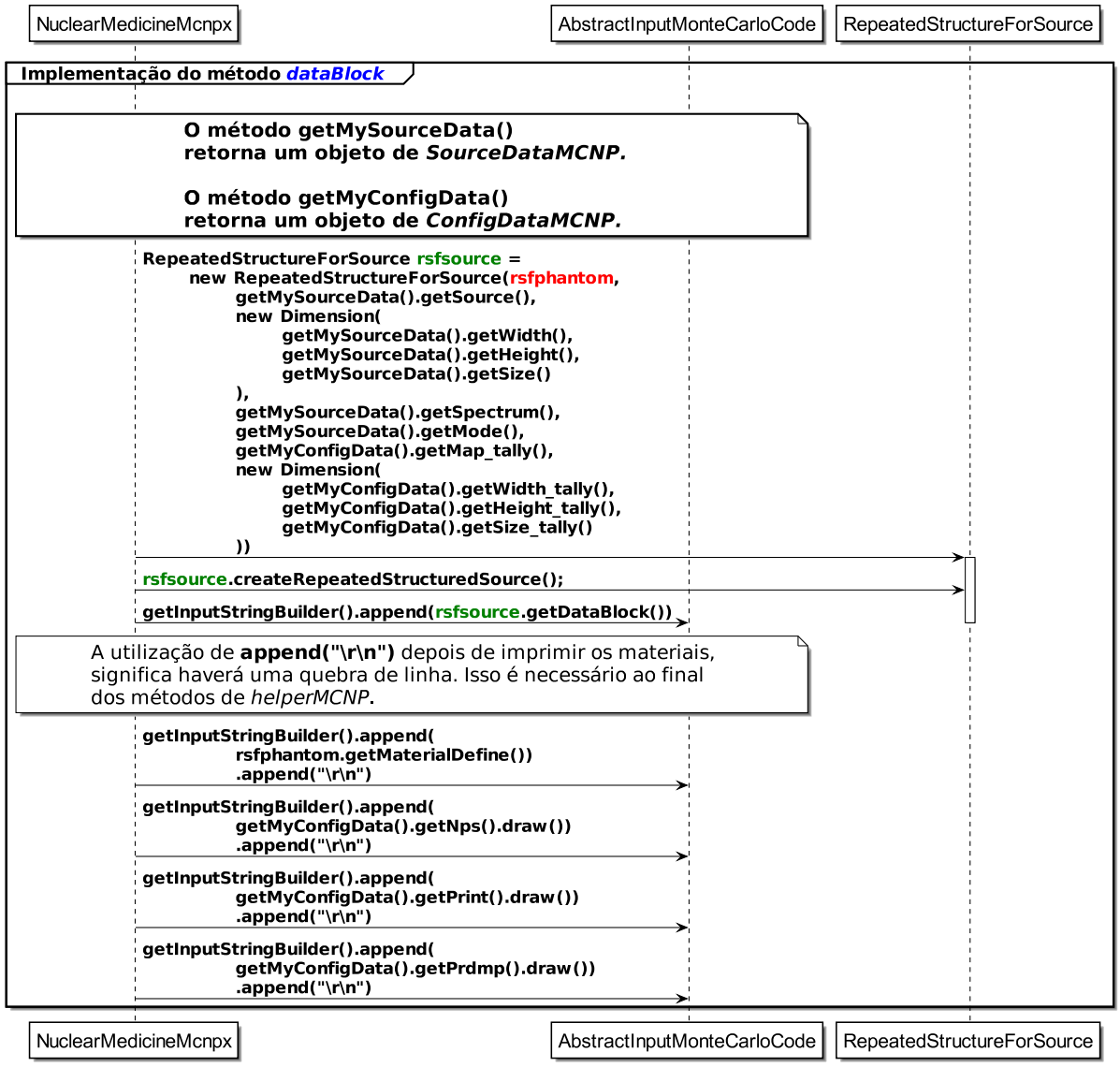

Figura 5.15: Diagrama de sequência da implementação do método *dataBlock*.

O sexto método implementado foi o *commentBlock*. Neste método é possível colocar informações extras que não serão processadas pelo *MCNP*. A seguir, na Figura 5.16, há um diagrama de sequência com a implementação desse método.

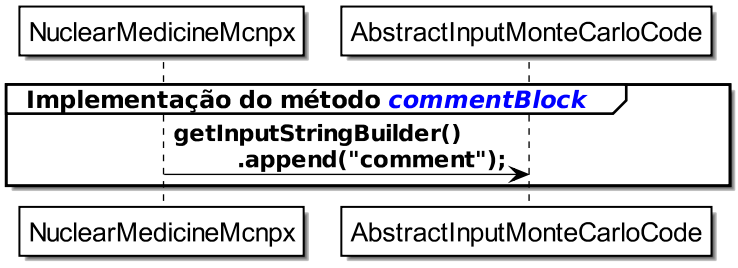

Figura 5.16: Diagrama de sequência da implementação do método *commentBlock*.

Simulação e comparação dos resultados com o software AMIGO: Em relação ao IBMC, após a elaboração dos módulos de dados e de construção foi possível gerar o arquivo *INP* do *MCNP*. A seguir encontra-se um diagrama de sequência, na Figura 5.17 com os passos seguidos para gerar esse arquivo.

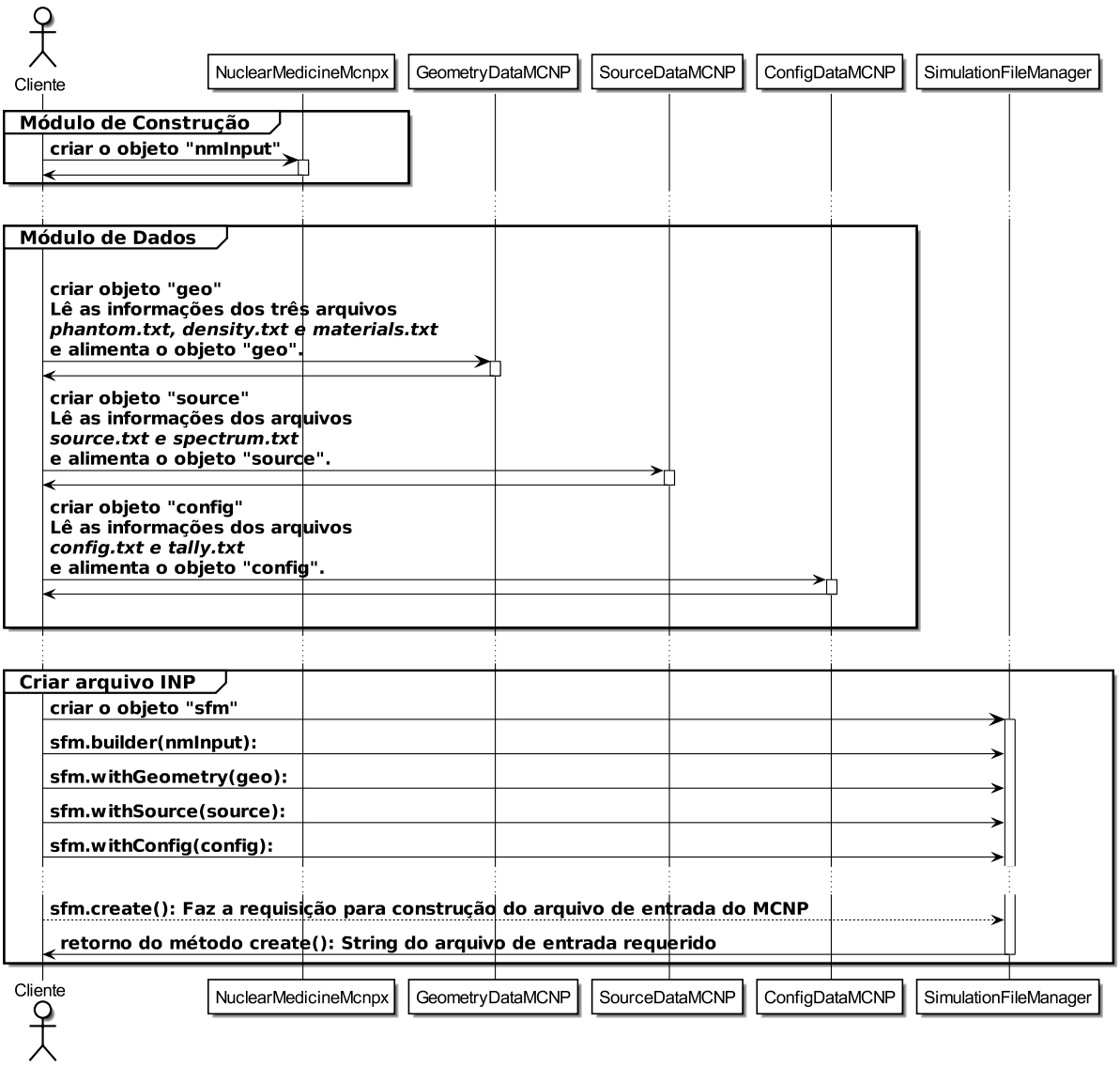

Figura 5.17: Diagrama de sequência que representa os passos para gerar o arquivo *INP* do *MCNP*, com o IBMC, para o caso de Medicina Nuclear.

As mesmas informações adquiridas pelo IBMC foram alimentadas no *software AMIGO* e então, os resultados das duas simulações foram comparados. Não houveram diferenças entre os resultados gerados pelo arquivo de entrada do *AMIGO* e os gerados pelo do IBMC.

Portanto, pode-se dizer que o IBMC possui a capacidade de simular um mesmo caso abordado pelo *AMIGO*.

Embora o processo de criação do arquivo *INP* a partir do IBMC pareça demorado, ele

se tornará cada vez mais rápido e prático. Isto porque, o IBMC proporciona um ambiente de desenvolvimento que facilita a reutilização de códigos. Isto será melhor entendido nos próximos casos.

5.1.2 Braquiterapia

Os procedimentos de braquiterapia são realizados por intermédio de fontes seladas, colocadas próximas ou em contato com o tumor com o auxílio de aplicadores. Para avaliar o uso do IBMC em casos de Braquiterapia, foi especificado um caso clínico de tratamento ginecológico intracavitário.

Este caso foi primeiramente simulado no *MCNP* pelo *software AMIGOBrachy* e depois simulado com o uso do IBMC[3]. Os resultados gerados pelas duas simulações foram comparados tendo como referência os resultados do *software AMIGOBrachy*.

A seguir, encontra-se a descrição deste caso.

- **Geometria:** Para especificar a geometria foram utilizadas 72 imagens com dimensões de 512 *voxels* de largura por 512 *voxels* de altura e 1 *voxel* de profundidade. As dimensões de cada *voxel* em centímetros são de 0,09765625 de largura, 0,09765625 de altura e 0,3 de profundidade. Essas imagens foram fornecidas em formato *DICOM*. Porém, o que foi simulado pelo *AMIGOBrachy* foi somente uma região de interesse com o objetivo tanto de aumentar a velocidade da simulação como de somente simular a região crítica para o tratamento. Dessa forma a nova dimensão foi de 470 *voxels* de largura e 373 *voxels* de altura, situados na mesma posição para todas as 72 imagens. A posição do *voxel* inicial, referente à essa região de interesse, foi no 30º *voxel* na largura e 60º na altura. Esse *voxel* se encontra na parte superior esquerda da imagem. Para visualizar melhor, vide Figura 5.18. Os tecidos foram segmentados pelo *AMIGOBrachy*: osso, músculo, tecido adiposo e ar mediante a escala de *Hounsfield*. O *AMIGOBrachy* também aplicou uma densidade para cada um desses tecidos. Vide a Tabela 5.7, para verificar as composições de cada tecido utilizado juntamente com suas densidades. O número, indicado ao lado dos materiais, na Tabela 5.7, correspondem aos valores no mapa de tecido. Todos esses dados de composições foram extraídos do *AMIGOBrachy*.

[3] <https://bitbucket.org/fmassicano/examples-of-ibmc/src/4fd7d2e8b14d9ee7db5710015f53e4d00f692c1b/src/main/java/examples/brachytherapy/?at=master>

- **Fonte de Radiação:** Foi utilizada um fonte selada de ^{192}Ir. Foram simuladas 17 posições de parada dessa fonte dentro do aplicador. O material do aplicador foi considerado na simulação como sendo de água. O tempo de parada e as posições das sementes dentro do aplicador foram fornecidas pelo *software AMIGOBrachy* que por sua vez extraiu essas informações do sistema de planejamento.

- ***Tally*:** A fim de especificar o *Tally* foi utilizado o cartão *FMESH* do *MCNP*. A região considerada foi de 366 *voxels* de comprimento por 173 *voxels* de altura para todas as 72 imagens. A posição do *voxel* inicial foi situada no 65º na largura e no 130º na altura. Esse *voxel* se encontra na parte inferior esquerda dessa região de interesse. Vide Figura 5.18 para visualizar tal região. Juntamente com o *FMESH* foi utilizado o cartão *DE* e *DF* a fim de caracterizar o coeficiente de atenuação da água em função da energia. Além disso, foram utilizados os cartões *PHYS* e *CUT* para fótons.

- **Configurações do *MCNP*:** Em relação as configurações do *MCNP*, foram utilizados os cartões *PRINT*, *DBCN*, *PRDMP* e *NPS*. Todas essas informações foram extraídas do *software AMIGOBrachy*.

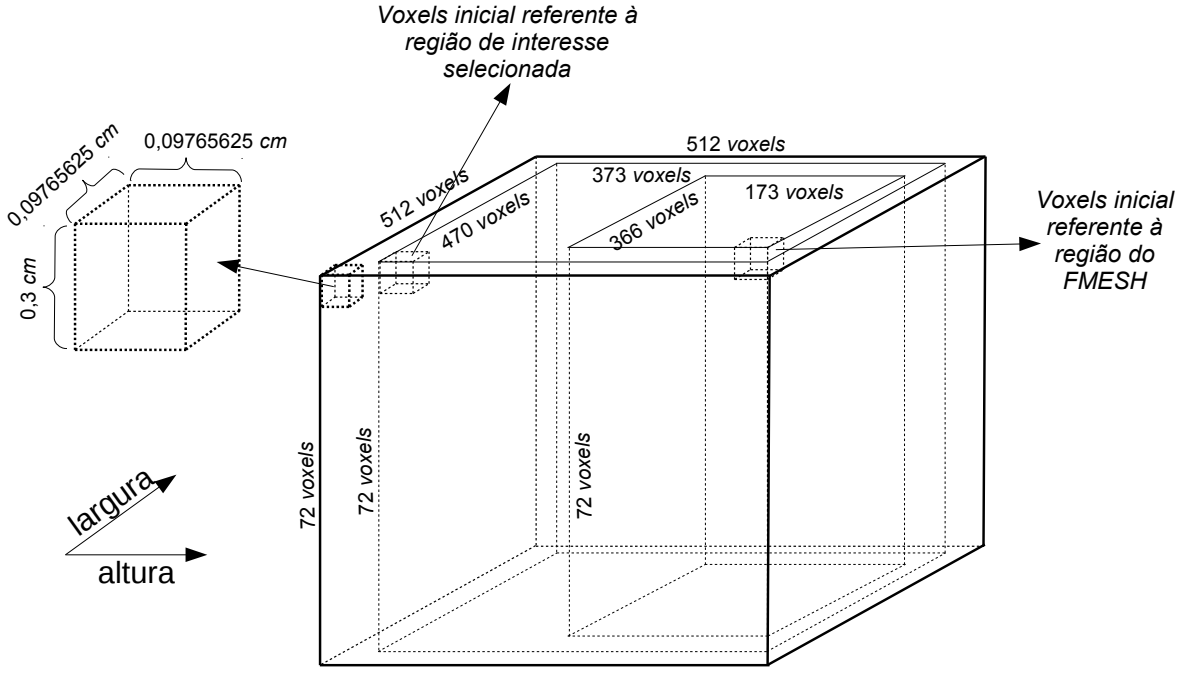

Figura 5.18: Representação da geometria de Braquiterapia.

Tabela 5.7 - Composição dos materiais referente à simulação de Braquiterapia.

		Composição Química (% em massa)				
Material (número)	ρ(g/cm^3)	H	C	N	O	Z>8
Ar (1)	0,00121	-	0,0124	75,527	23,178	Ar(1,2827)
Tecido Adiposo (3)	0,92	11,9477	64,7240	0,7970	23,2333	Na(0,05), Mg(0,002), P(0,016), S(0,073), Cl(0,119), K(0,032), Ca(0,002), Fe(0,002), Zn(0,002)
Osso (11)	1,85	4,7234	14,4330	4,1990	44,6096	Mg(0,2200), P(10,497), S(0,315), Ca(20,993), Zn(0,01)
Músculo (13)	1,05	10,0637	10,7830	2,7680	75,4773	Na(0,075), Mg(0,0190), P(0,18), S(0,241), Cl(0,079), K(0,302), Ca(0,003), Fe(0,004), Zn(0,005)

Nesse caso, para simular o tratamento o *software AMIGOBrachy* extrai as informações do sistema de planejamento de Braquiterapia. Já o sistema que foi desenvolvido com o IBMC extrai as informações do *software AMIGO*. Na Figura 5.19, há um esquema do fluxo de dados para a simulação desse caso, com os *softwares AMIGOBrachy* e IBMC.

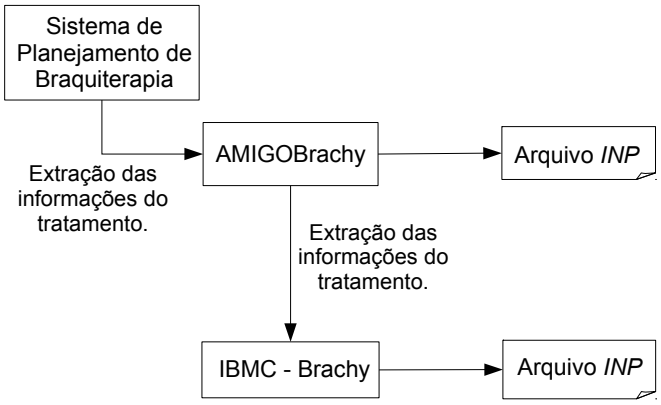

Figura 5.19: Fluxo de dados para a criação dos arquivos do *MCNP* com o uso do *software AMIGOBrachy* e o IBMC.

A seguir serão descritos os passos para gerar o arquivo *INP* pelo IBMC.

Criar Módulo de Dados referente à Geometria: Para extrair as informações do *software AMIGOBrachy* foi utilizada a classe *GeometryDataMCNP*, já criada no caso anterior de Medicina Nuclear, com alguns métodos adicionados para lidar com esse caso em específico.

O *AMIGOBrachy* forneceu a geometria através de um arquivo denominado *materials.txt*. Este arquivo contem matrizes com os identificadores do tecido. Ele forma esse

arquivo mediante as imagens *DICOM* de Tomografia Computadorizada. Dentro destes arquivos as matrizes se encontram uma abaixo da outra.

As informações das composições e densidades dos materiais, região de interesse e tamanho do *voxel* fornecidas pelo *AMIGOBrachy* foram cadastradas manualmente no objeto da classe *GeometryDataMCNP*. Esse processo é observado na Figura 5.21.

A Figura 5.20, mostra um diagrama de classe correspondente a classe *GeometryDataMCNP*. Somente foram apresentados os métodos utilizados nesse caso. Em verde encontram-se os novos métodos que foram implementados.

Foram adicionados somente dois métodos. O primeiro é o *setInicialPoint* que recebe como parâmetro um objeto da classe *Point* o qual na Figura 5.18 representa o *voxel* inicial referente à região de interesse selecionada. O segundo método foi o *setMapROI* que recebe três números inteiros os quais representam as dimensões dessa região de interesse.

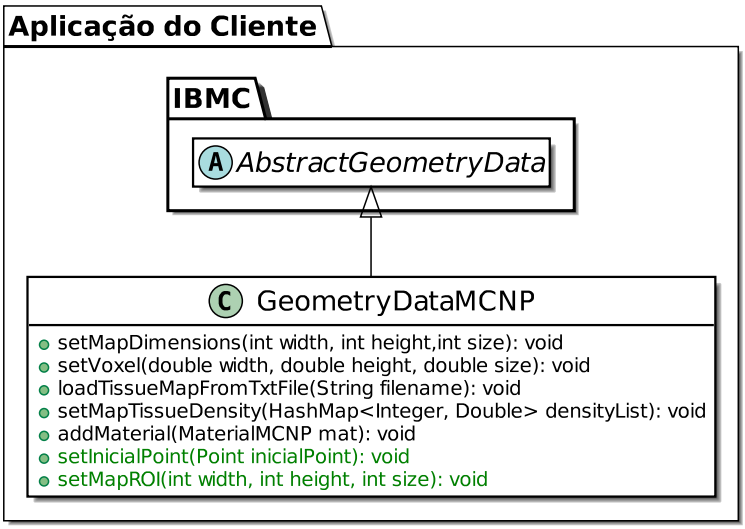

Figura 5.20: Diagrama de classes de *GeometryDataMCNP* para o caso de Braquiterapia.

Na Figura 5.21 há um diagrama de sequência que mostra como o objeto de *GeometryDataMCNP* foi preenchido.

Primeiramente é criado o objeto ***geometryData***. Em seguidas foram especificadas as dimensões dos *voxels*, das matrizes contendo os identificadores dos tecidos e da região de interesse. Depois as matrizes foram armazenadas por meio do método *loadTissueMapFromTxtFile*.

As composições dos tecidos e suas densidades foram especificadas utilizando o método *addMaterial* e o *setMapTissueDensity*.

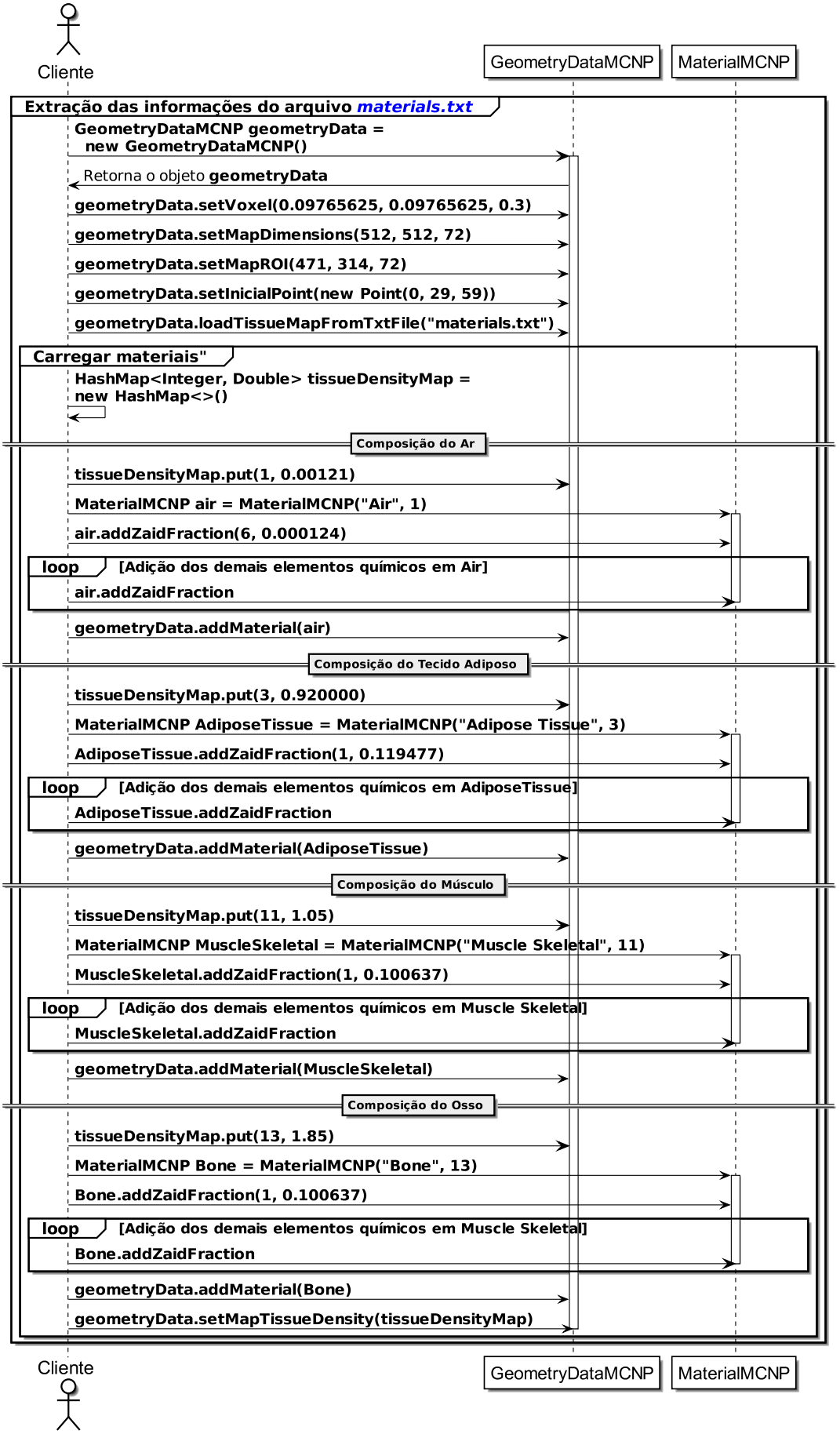

Figura 5.21: Diagrama de sequência do preenchimento de um objeto da classe *GeometryDataMCNP* para o caso de Braquiterapia.

Criar Módulo de Dados referente à Fonte de Radiação: Para extrair as informações da fonte de radiação do *software AMIGOBrachy* foi utilizada a classe *SourceDataMCNP*, já criada no caso anterior de Medicina Nuclear. Para lidar com esse caso em específico foram adicionados alguns métodos em *SourceDataMCNP*.

A Figura 5.22 mostra o diagrama de classes de *SourceDataMCNP*. Os métodos novos que foram adicionados estão em verde na Figura 5.22.

Para representar a fonte de radiação o *AMIGOBrachy* utilizou a cartão *SSR*. Para inclusão desse cartão em *SourceDataMCNP* foi criado o método *setSRR*. Quando utilizado, o cartão *SSR*, exige que sejam especificadas as superfícies da fonte de radiação simulada com o cartão *SSW*. Dessa forma, criou-se o método *addSurf*.

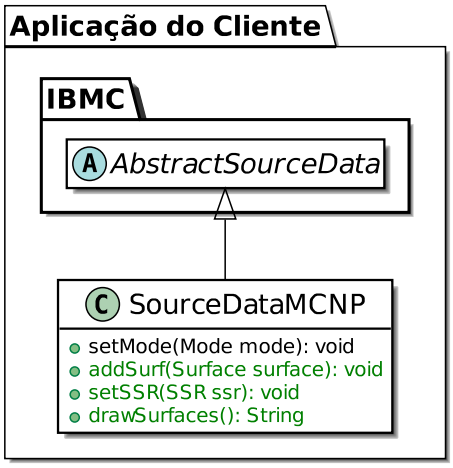

Figura 5.22: Diagrama de classes de *GeometryDataMCNP* para o caso de Braquiterapia.

A Figura 5.23 mostra um diagrama de sequência que mostra como o objeto de *SourceDataMCNP* foi preenchido.

Primeiramente foi criado o objeto ***sd*** da classe *SourceDataMCNP*. Em seguida, foi especificado o cartão *Mode* através do método *setMode*. Depois disso, foram especificadas as superfícies da fonte de radiação que estavam contidas no cartão *SSW*.

Depois, houve a criação do objeto ***ssr*** da classe *SSR*. Nela, foram adicionadas, no parâmetro *OLD*, as superfícies mediante o método *withOLD* o qual recebe uma lista de superfícies. As posições da semente dentro do aplicador foram simuladas com o uso do cartão *TR* vinculado ao *SI* e *SP*. Por fim foi atribuido um valor ao parâmetro *WGT* do *SSR*. Todas essas informações não foram calculadas, mas sim foram provenientes do *AMIGOBrachy*. Por fim, após todo esse processamento, o objeto ***ssr*** foi adicionado em

sd.

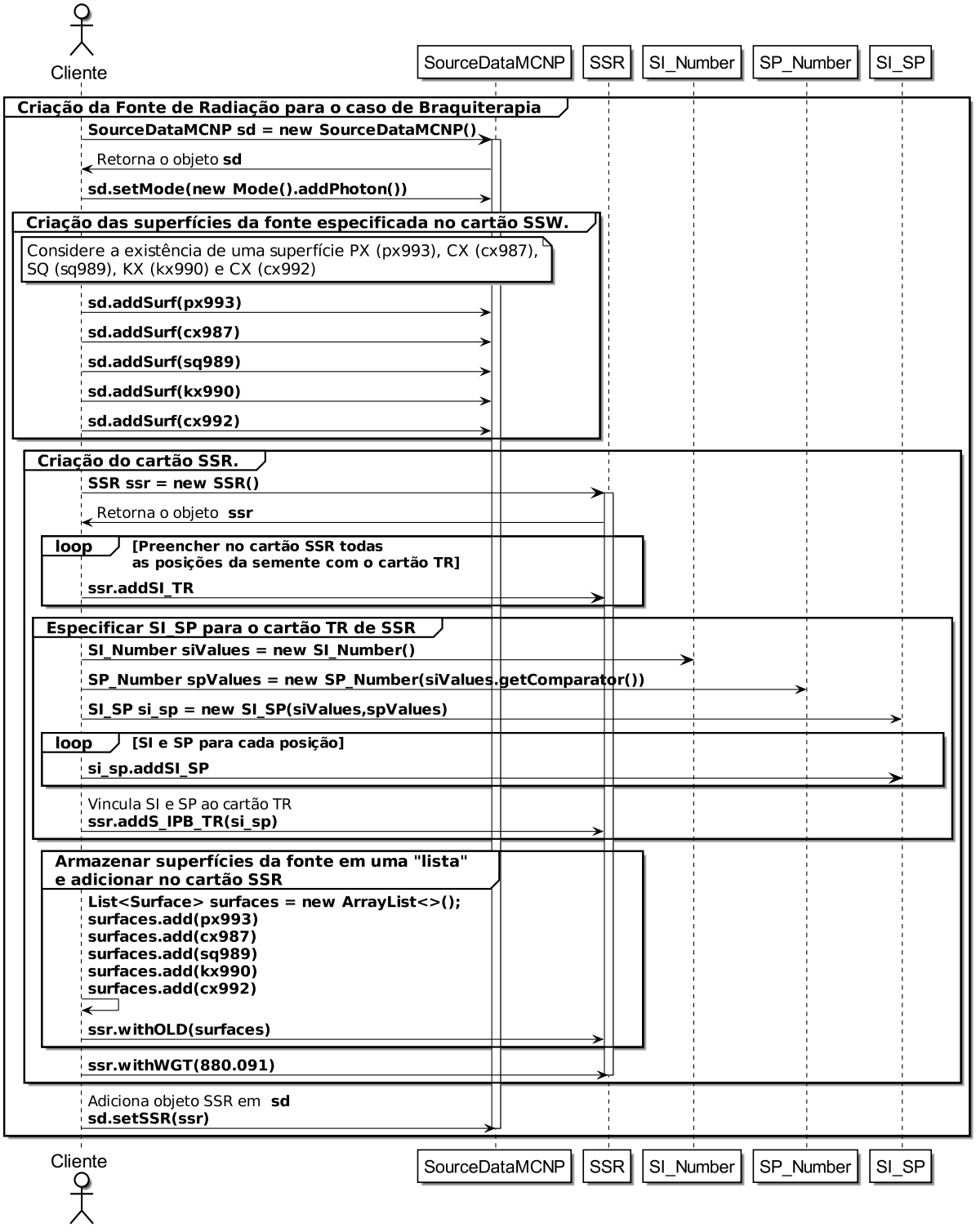

Figura 5.23: Diagrama de sequência do preenchimento de um objeto da classe *SourceDataMCNP* para o caso de Braquiterapia.

Criar Módulo de Dados referente à Configuração: A classe *ConfigDataMCNP*, usada no caso de Medicina Nuclear, foi utilizada para armazenar as informações corres-

pondentes à configuração do *MCNP*, provenientes do *software AMIGOBrachy*. Alguns métodos foram adicionados em *ConfigDataMCNP* a fim de contemplar esse caso.

A Figura 5.24 mostra o diagrama de classes de *ConfigDataMCNP*. Somente os métodos utilizados nesse caso foram apresentados, sendo que os métodos novos estão em verde.

Essa foi a única classe dos módulos de dados que nenhum método da classe foi aproveitado. Até poderia se ter criado outra classe para as configurações, porém optou-se por utilizar a *ConfigDataMCNP* implementando nela, funcionalidades novas.

Para esse caso de Braquiterapia, o *AMIGOBrachy* criou o *Tally* com o *FMESH*, então foi implementado o método *setFMESH* em *ConfigDataMCNP*. Também foram adicionados os métodos *setDbcn*, para o cartão *DBCN*, *setPhysCard*, para o *PHYS*, *setCutCard*, para o *CUT*, *setPRDMP* para o *PRDMP*, *setNps* para o *NPS* e por fim o *setPRINT* para o cartão *PRINT*.

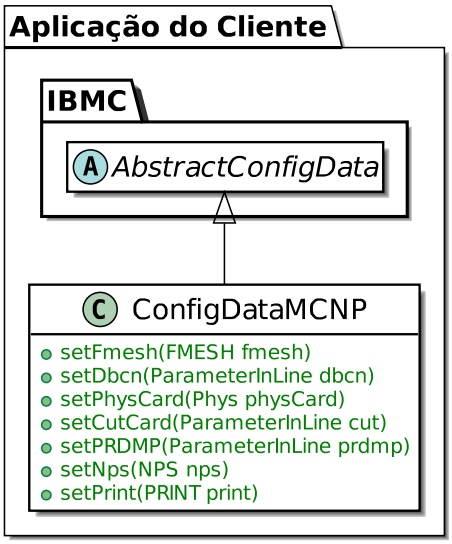

Figura 5.24: Diagrama de classes de *ConfigDataMCNP* para o caso de Braquiterapia.

A Figura 5.25 mostra o diagrama de sequência que apresenta o preenchimento de um objeto da classe *ConfigDataMCNP*. Primeiramente foi criado o objeto ***configData*** da classe *ConfigDataMCNP*. Nos passos seguintes foram criados os objetos referentes aos cartões *FMESH*, *PHYS*, *CUT*, *DBCN*, *NPS* e *PRINT*. Por fim, esses objetos foram adicionados em ***configData***.

Todos os parâmetros, para formação dos objetos correspondentes aos cartões do *MCNP*, foram extraídos do *AMIGOBrachy*.

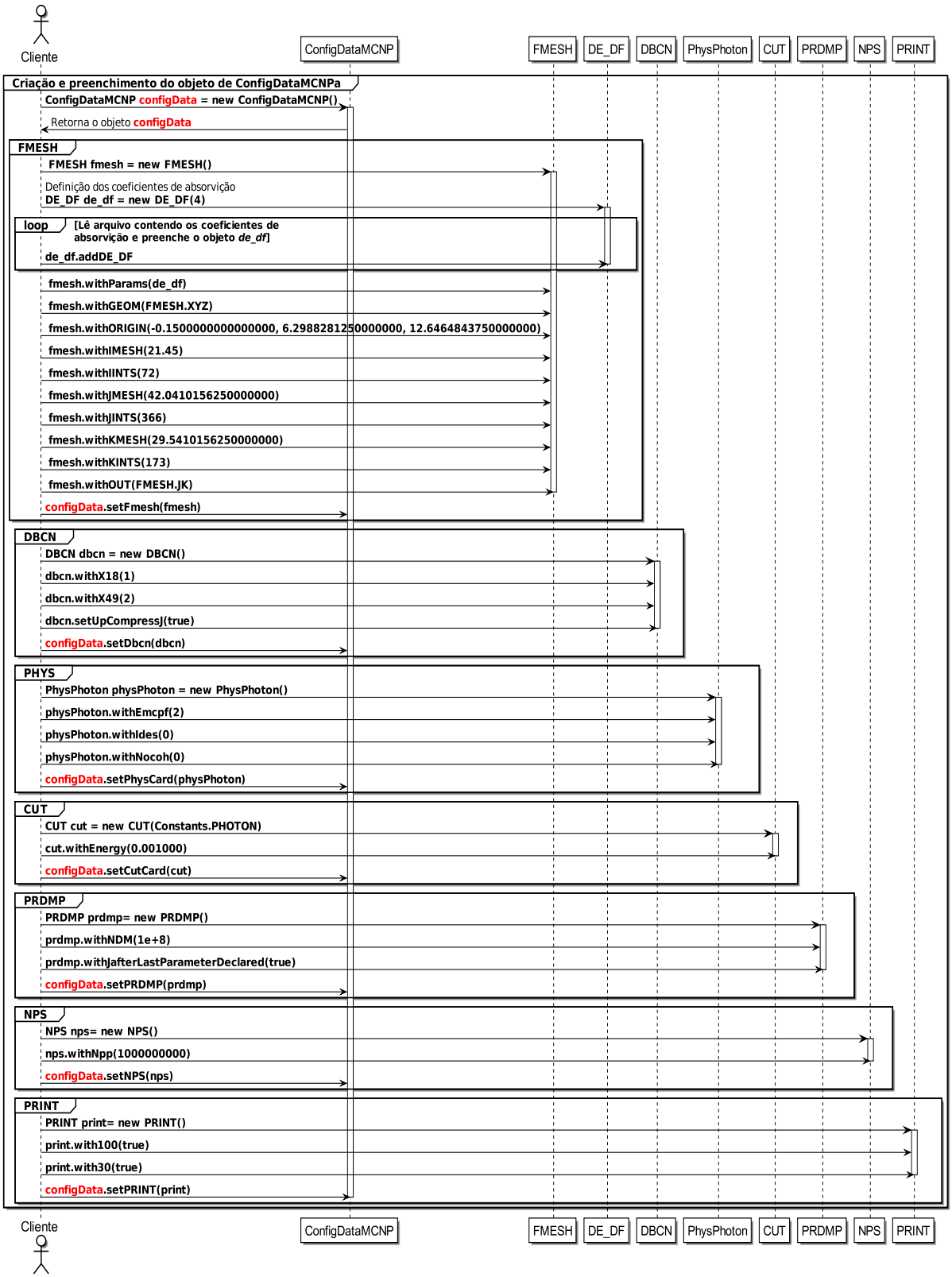

Figura 5.25: Diagrama de sequência do preenchimento de um objeto da classe *ConfigDataMCNP* para o caso de Braquiterapia.

Criar Módulo de Construção: A classe referente ao Módulo de Construção foi denominada *BrachytherapySTP*. Para auxiliá-la na construção do arquivo *INP* do *MCNP* foi

utilizada a classe *RepeatedStructureForPhantom*.

A Figura 5.26 mostra um diagrama de sequência que apresenta as implementações os principais métodos de *BrachytherapySTP*.

No método *cellBlock* foi utilizada a classe *RepeatedStructureForPhantom* para construir o objeto simulador de *voxels* no *MCNP*.

O método *surfaceBlock* além das superfícies estabelecidas pela estrutura repetidas contêm as superfícies da fonte de radiação acessada pelo método *drawSurfaces* da classe *SourceDataMCNP*.

Por fim, no método *dataBlock* ocorre a impressão de todos os cartões especificados em *ConfigDataMCNP* destinados a esse caso de Braquiterapia.

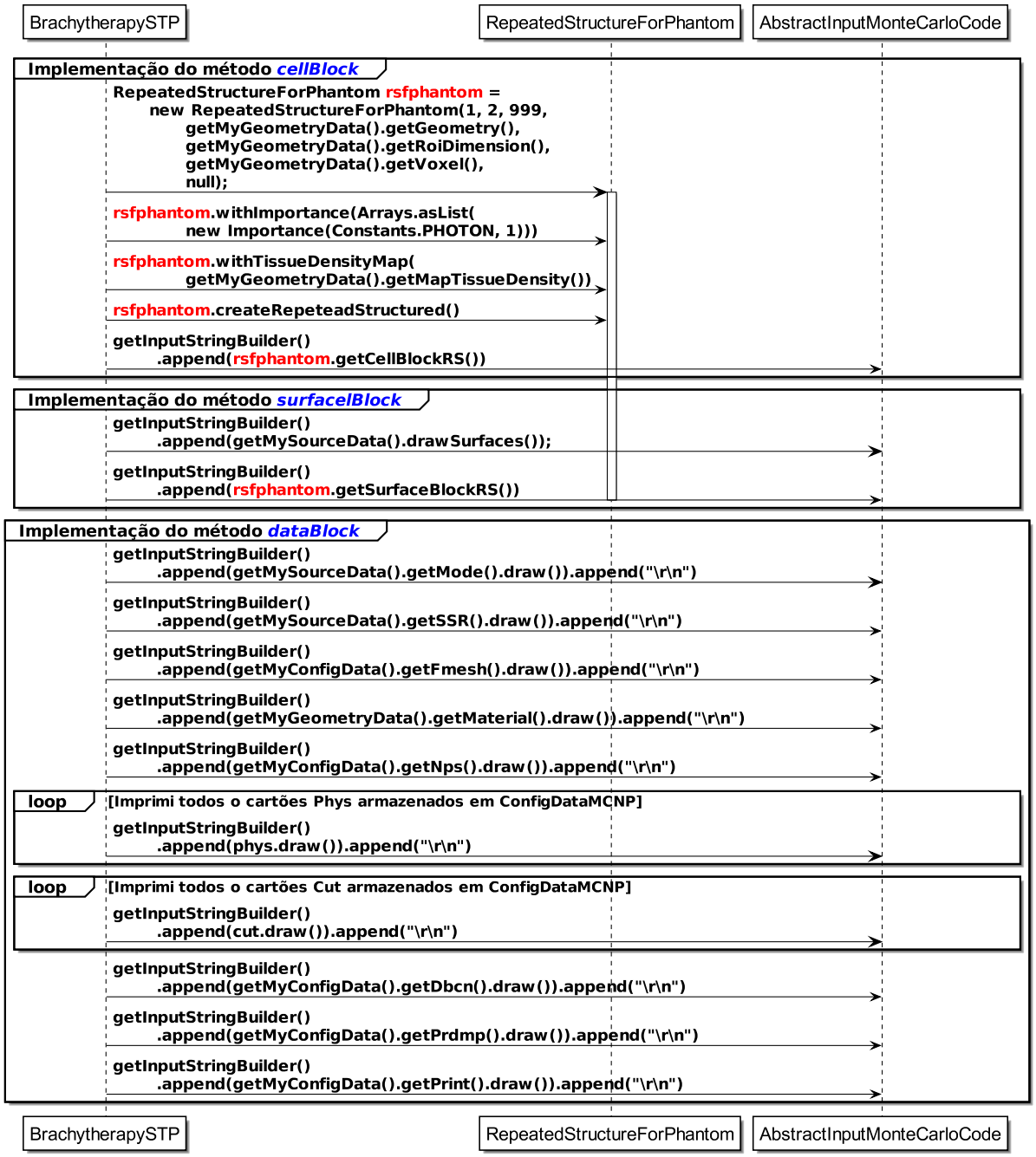

Figura 5.26: Diagrama de sequência referente à criação do arquivo de entrada para o *MCNP* para o caso de Braquiterapia.

A Figura 5.27 mostra a geometria do paciente desse caso de Braquiterapia. Na Figura 5.27(a) é apresentado um corte transversal onde é possível ver uma região retangular em vermelho, essa região representa a localização do *Tally*. Figura 5.27(b) é apresentado um corte coronal onde é possível identificar o aplicador.

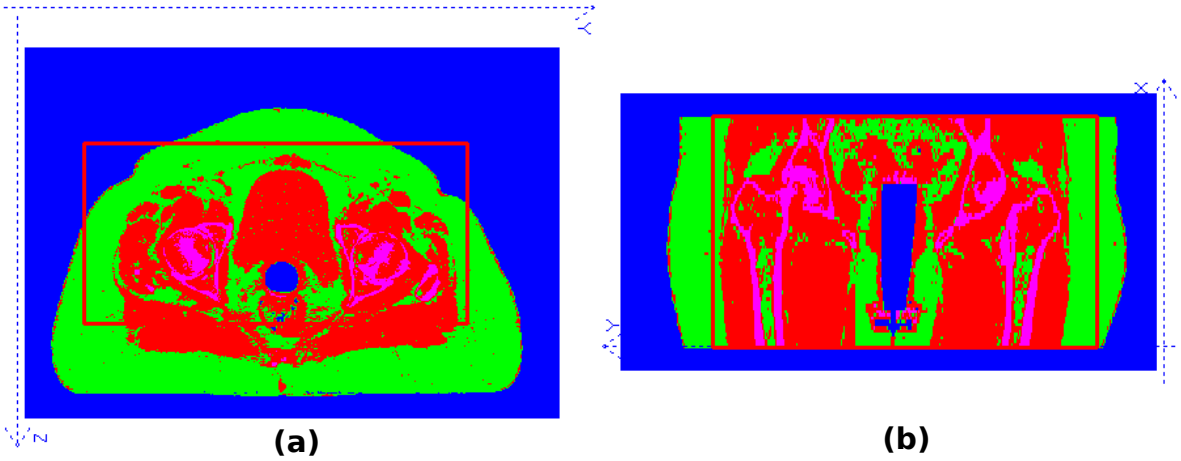

Figura 5.27: Imagem da geometria do paciente, no caso de Braquiterapia, criado com o auxílio do IBMC. Em (a) tem-se um corte transversal do paciente e em (b) um corte coronal.

Não houveram diferenças nos resultados gerados pelos arquivos *INP* do *AMIGOBrachy* e do IBMC. Isso indica que o IBMC conseguiu reproduzir os mesmos resultados gerados pelo *AMIGOBrachy*. Outro fato importante foi que o IBMC propiciou a criação de um caso de Braquiterapia com a utilização de uma base de dados externas, *AMIGOBrachy*, de maneira rápida e flexível. Sendo que um dos principais motivos para isso ter ocorrido foi a reutilização de códigos, como foi o caso das classes *GeometryDataMCNP*, *SourceDataMCNP* e *RepeatedStructureForPhantom*.

5.1.3 Teleterapia

A Teleterapia é uma modalidade da Radioterapia em que os tratamentos são realizados mediante fontes de radiação externas colocadas a uma certa distância do paciente, normalmente igual ao isocentro do equipamento de teleterapia, no caso dos aceleradores lineares esta distância equivale a 1,0 metro. Para avaliar o uso do IBMC em casos de Teleterapia, foi especificado um caso.

Este caso foi primeiramente simulado no *MCNP* pelo *software AMIGOBrachy* e depois simulado com o uso do IBMC[4]. Os resultados gerados pelas duas simulações foram comparados tendo como referência os resultados do *software AMIGOBrachy*.

A seguir, encontra-se a descrição deste caso.

- **Geometria:** Para especificar a geometria foram utilizadas 315 imagens com dimensões de 512 *voxels* de largura por 512 *voxels* de altura e 1 *voxel* de profundidade. As

[4] <https://bitbucket.org/fmassicano/examples-of-ibmc/src/4fd7d2e8b14d9ee7db5710015f53e4d00f692c1b/src/main/java/examples/radiotherapy/?at=master>

dimensões de cada *voxel* em centímetros são de 0.078125 de largura e de altura e 0,1 centímetros de profundidade. O *AMIGOBrachy* somente simulou uma determinada região de interesse com dimensões de 512 *voxels* de largura por 301 *voxels* de altura, situados na mesma posição para todas as 315 imagens. A posição do *voxel* inicial, referente à essa região de interesse, foi no 100º *voxel* da altura. Esse *voxel* se encontra na parte superior esquerda da imagem. Para visualizar melhor, vide Figura 5.28. Os tecidos foram segmentados pelo *AMIGOBrachy* osso, músculo, tecido adiposo e ar mediante a escala de *Hounsfield*. O *AMIGOBrachy* também aplicou uma densidade para cada um desses tecidos. Vide a Tabela 5.8, para verificar as composições de cada tecido utilizado juntamente com suas densidades.

- **Fonte de Radiação:** Foi estabelecida uma fonte retangular plana de fótons, referente ao Cobalto-60, com dimensões de 4 centímetros de largura por 16 centímetros de altura situada a uma distância de 1 metro do paciente. Vide Figura 5.28 para visualizar a fonte.

- ***Tally*:** A fim de especificar o *Tally* foi utilizado o cartão *FMESH* do *MCNP*, com dimensões iguais a da região de interesse. Juntamente com o *FMESH* foi utilizado o cartão *FM*. Além disso, foram utilizados os cartões *PHYS* e *CUT* para fótons.

- **Configurações do *MCNP*:** Em relação as configurações do *MCNP*, foram utilizados os cartões *PRINT*, *DBCN*, *PRDMP* e *NPS*. Todas essas informações foram extraídas do *software AMIGOBrachy*.

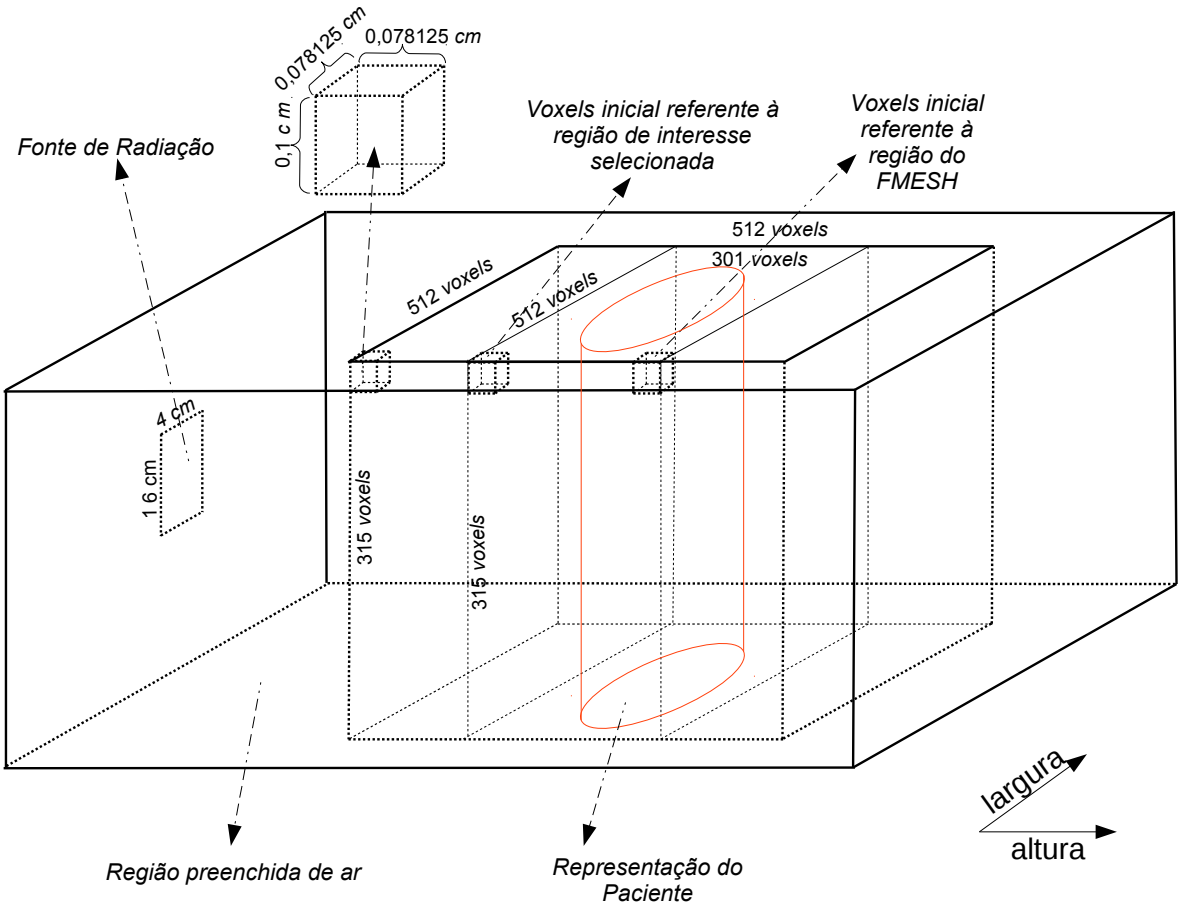

Figura 5.28: Representação da geometria de Teleterapia.

Tabela 5.8 - Composição dos materiais referente à simulação de Teleterapia.

		Composição Química (% em massa)				
Material (número)	ρ(g/cm^3)	H	C	N	O	Z>8
Ar (1)	0,00121	-	0,0124	75,527	23,178	Ar(1,2827)
Tecido Adiposo (3)	0,92	11,9477	64,7240	0,7970	23,2333	Na(0,05), Mg(0,002), P(0,016), S(0,073), Cl(0,119), K(0,032), Ca(0,002), Fe(0,002), Zn(0,002)
Osso (11)	1,85	4,7234	14,4330	4,1990	44,6096	Mg(0,2200), P(10,497), S(0,315), Ca(20,993), Zn(0,01)
Músculo (13)	1,05	10,0637	10,7830	2,7680	75,4773	Na(0,075), Mg(0,0190), P(0,18), S(0,241), Cl(0,079), K(0,302), Ca(0,003), Fe(0,004), Zn(0,005)

Nesse caso, para simular o tratamento o IBMC extrai as informações do *software AMIGO*. A seguir serão descritos os passos para gerar o arquivo *INP* pelo IBMC.

Criar Módulo de Dados referente à Geometria: Assim como nos dois casos anteriores, os dados da geometria foram armazenados em um objeto da classe *GeometryDataMCNP*.

A geometria, proveniente do *AMIGOBrachy*, foi armazenada na classe *GeometryDataMCNP* mediante a leitura de um arquivo texto, *Mat_HU.txt* produzido pelo *AMIGOBrachy*. Dentro deste arquivo há os identificadores do tecido extraídos por imagens de Tomografia Computadorizada. Informações de composição e densidade foram cadastradas manualmente no objeto da classe *GeometryDataMCNP*.

Não houveram modificações na classe *GeometryDataMCNP*, e na Figura 5.29, há um diagrama de sequência que mostra como o objeto de *GeometryDataMCNP* foi preenchido para esse caso.

O primeiro passo foi criar o objeto ***geometryData***. Depois esse objeto foi preenchido com as dimensões dos *voxels*, das matrizes contendo os identificadores dos tecidos e da região de interesse. Depois foi especificado o ponto inicial do *voxel* mediante a método *setInicialPoint*. As matrizes foram armazenadas por meio do método *loadTissueMapFromTxtFile*.

Por fim, as composições dos tecidos e suas densidades foram especificadas utilizando o método *addMaterial* e o *setMapTissueDensity*.

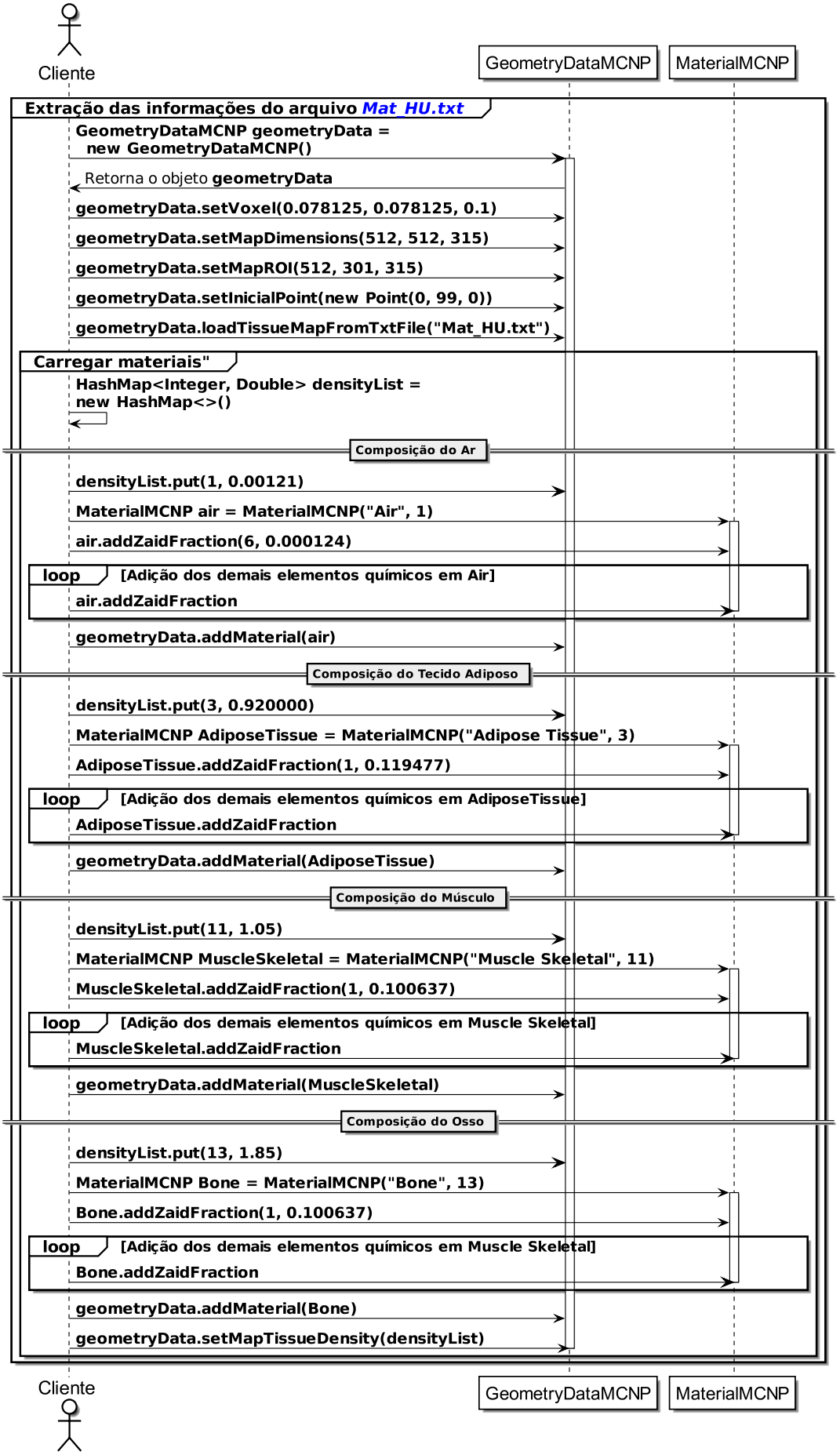

Figura 5.29: Diagrama de sequência do preenchimento de um objeto da classe *GeometryDataMCNP* para o caso de Teleterapia.

Criar Módulo de Dados referente à Fonte de Radiação: Para extrair as informações da fonte de radiação do *software AMIGOBrachy* foi utilizada a classe *SourceDataMCNP*, já criada no caso anterior de Medicina Nuclear e Braquiterapia. A classe *SourceDataMCNP* não precisou ser modificada.

Para representar a fonte de radiação o *AMIGOBrachy* utilizou a cartão *SDEF*. Para inclusão desse cartão em *SourceDataMCNP* é especificada pelo método *setSdef*.

Na Figura 5.30 há um diagrama de sequência que mostra como o objeto de *SourceDataMCNP* foi preenchido.

Primeiramente foi criado o objeto ***sourceData*** da classe *SourceDataMCNP*. Em seguida, foi especificado o cartão *Mode* através do método *setMode*. Por fim, foram especificados os parâmetros do cartão *SDEF* e adicionados no objeto *sourceData*.

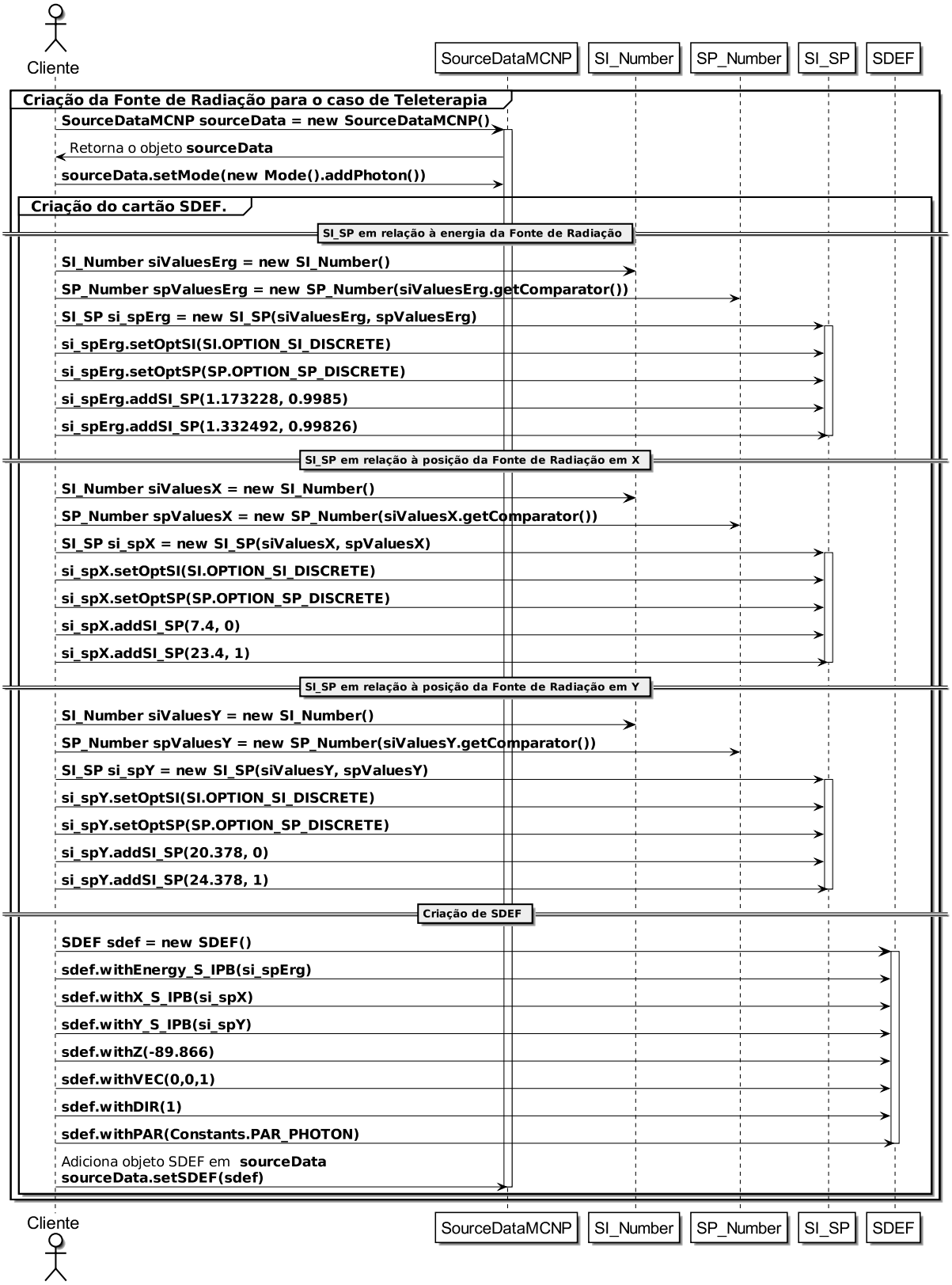

Figura 5.30: Diagrama de sequência do preenchimento de um objeto da classe *SourceDataMCNP* para o caso de Teleterapia.

Criar Módulo de Dados referente à Configuração: Da mesma forma como nos casos de Medicina Nuclear e Braquiterapia, para esse caso foi utilizada a classe *ConfigDataMCNP* para armazenar as informações correspondentes à configuração do *MCNP*. Essas informações foram provenientes do *software AMIGOBrachy*.

Na Figura 5.31 há o diagrama de sequência que apresenta o preenchimento de um objeto da classe *ConfigDataMCNP*. Primeiramente foi criado o objeto ***configData*** da classe *ConfigDataMCNP*. A seguir é criado e preenchido o objeto de *FMESH*. Depois são especificados os objetos de *PHYS*, *CUT*, *DBCN*, *NPS* e *PRINT*. Por fim, esses objetos foram adicionados em ***configData***.

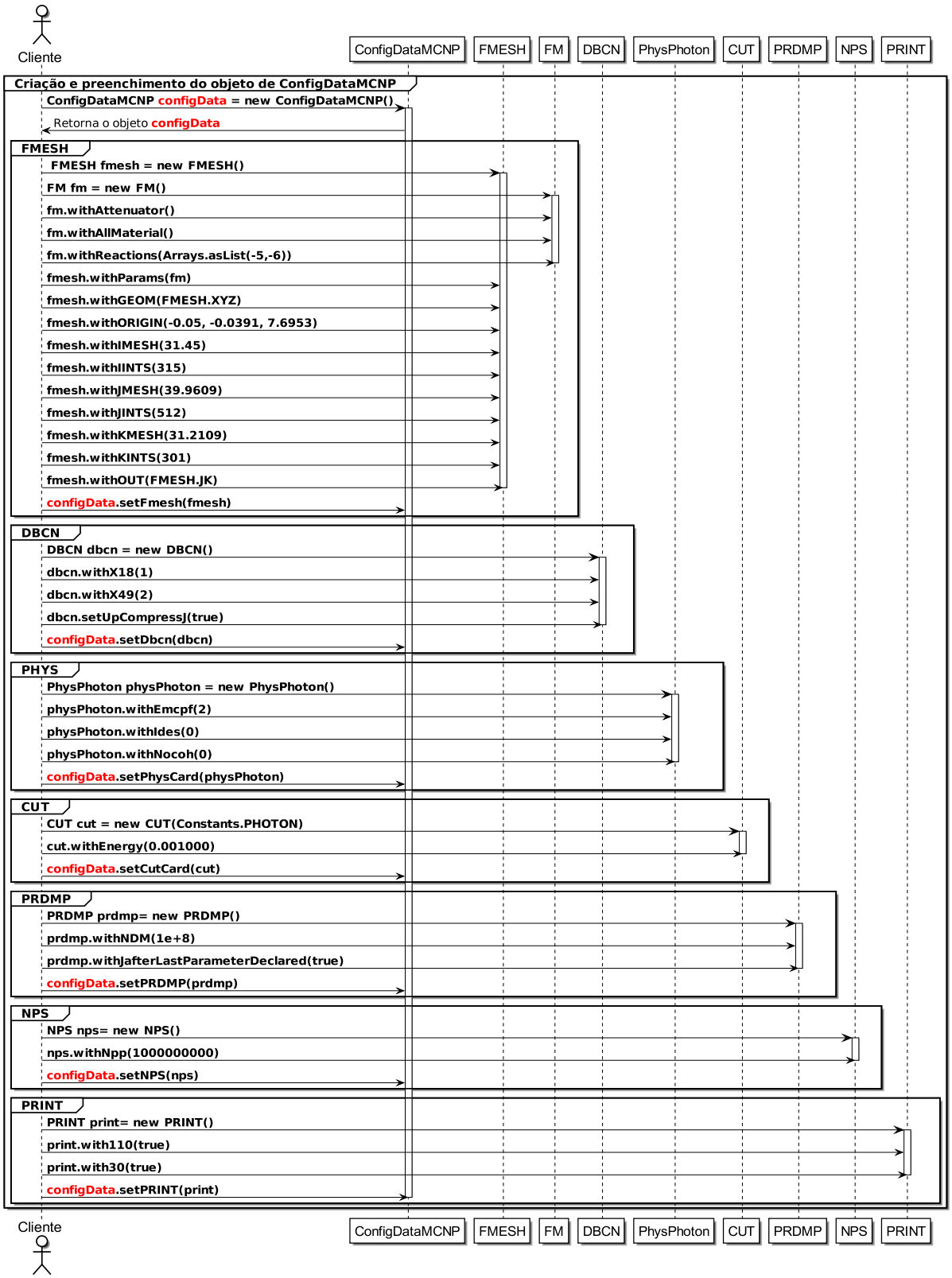

Figura 5.31: Diagrama de sequência do preenchimento de um objeto da classe *ConfigDataMCNP* para o caso de Teleterapia.

Criar Módulo de Construção: A classe referente ao Módulo de Construção foi denominada *RadiotherapySTP*. Para auxiliá-la na construção do arquivo *INP* do *MCNP* foi

utilizada a classe *RepeatedStructureForPhantom*.

A fim de contemplar esse caso, foi adicionado alguns métodos na classe *RepeatedStructureForPhantom*. O principal método adicionado foi o *dimAir*. Esse método foi criado pois ao redor do objeto simulador de *voxels* foi colocada uma camada de ar. Anteriormente essa camada era definida a dois centímetros de distância do objeto simulador. Nos dois casos anteriores as fontes ou se encontravam dentro do objeto simulador ou muito próximas destes. Nesse caso em específico a fonte é externa ao objeto simulador, a uma distância de um metro, e essa camada de ar deverá ser estendida a fim de contemplar a fonte de radiação mais o objeto simulador. Isso pode ser observado na Figura 5.28.

A Figura 5.32 ilustra um diagrama de sequência que apresenta as implementações dos principais métodos de *RadiotherapySTP*.

No método *cellBlock* foi utilizada a classe *RepeatedStructureForPhantom* para construir o objeto simulador de *voxels* no *MCNP*.

O método *surfaceBlock* além das superfícies estabelecidas pela estrutura repetidas contêm as superfícies da fonte de radiação acessada pelo método *drawSurfaces* da classe *SourceDataMCNP*.

Por fim, no método *dataBlock* ocorre a impressão de todos os cartões especificados em *ConfigDataMCNP* destinados a esse caso de Teleterapia.

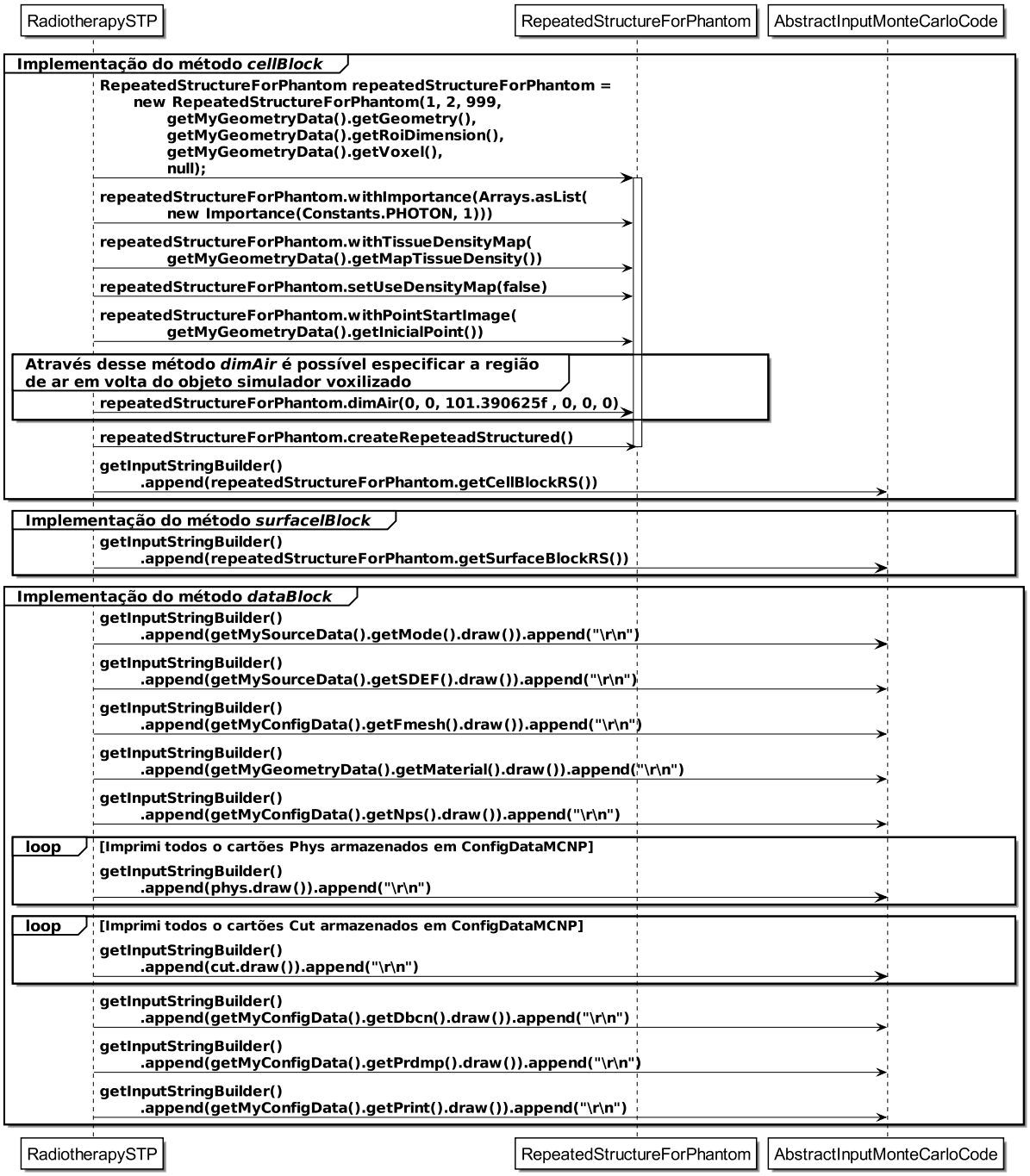

Figura 5.32: Diagrama de sequência referente à construção do arquivo do MCNP para o caso de Teleterapia.

A Figura 5.33 apresenta a representação da geometria do paciente para o caso de Teleterapia. Na Figura 5.33(a) é apresentado um corte axial do paciente juntamente com a presença da camada de ar que envolve o paciente e a fonte de radiação. Na Figura 5.33(b) é apresentado um corte coronal contendo a delimitação da região irradiada.

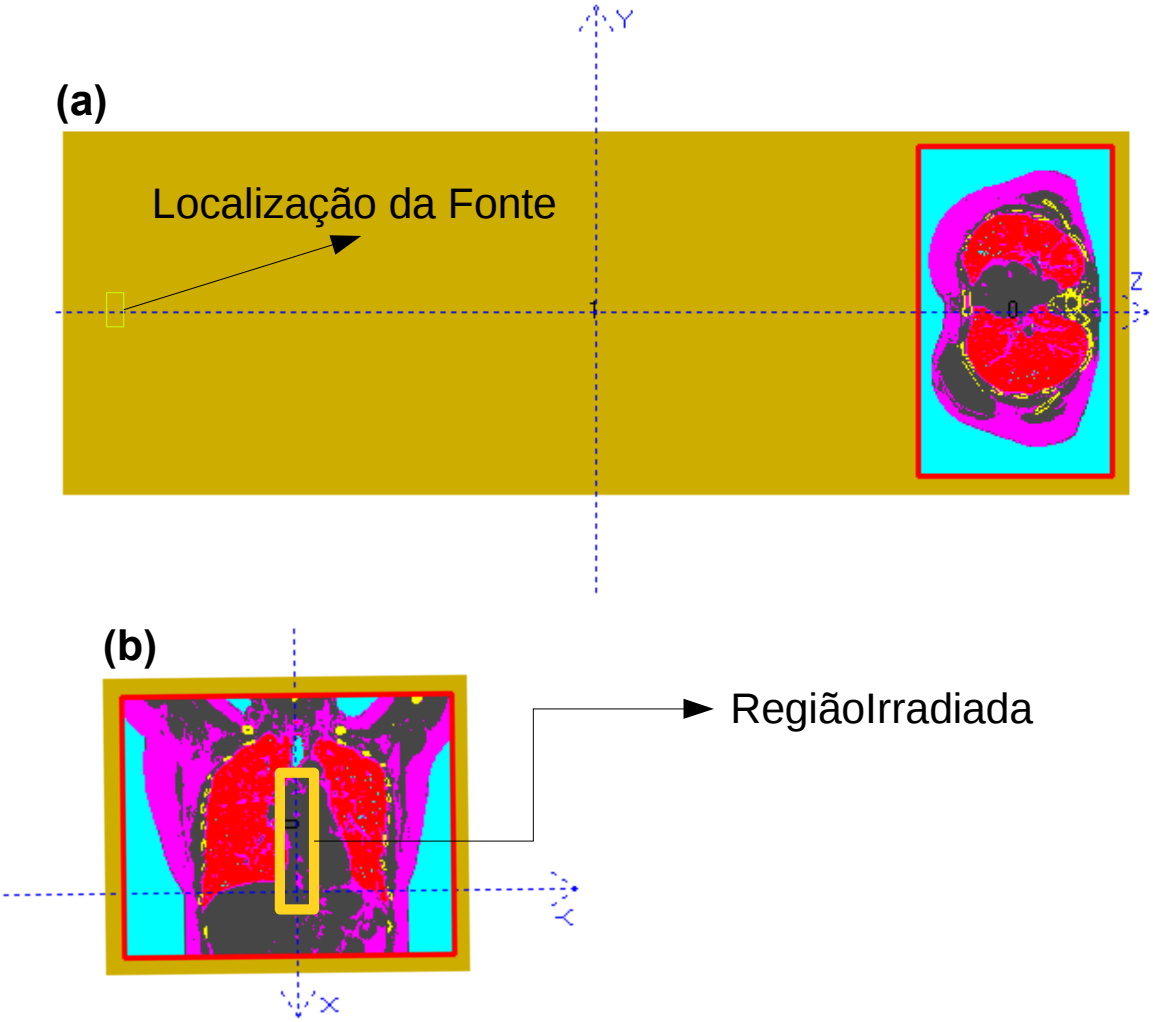

Figura 5.33: Geometria do paciente, criada com o auxílio do IBMC, para o caso de Teleterapia. Em (a) é apresentado um corte transversal do paciente e em (b) um corte transversal.

Para esse caso, também não houveram diferenças nos resultados gerados pelos arquivos *INP* do *AMIGOBrachy* e do IBMC. Isso indica que o IBMC conseguiu reproduzir os mesmos resultados gerados pelo *AMIGOBrachy*. Assim como no caso de Braquiterapia o IBMC propiciou a criação de um caso de Teleterapia com a utilização de uma base de dados externas, *AMIGOBrachy*, de maneira rápida e flexível. Isso reforça a grande importância da reutilização de códigos para que o processo de desenvolvimento de novos casos, com o uso do IBMC, seja feita de maneira rápida.

5.1.4 *Exemplo de uma possível integração de IBMC com o ICCT a fim de gerar a geometria de um paciente*

Como já mencionado na seção 4.1.4 o *ICCT* é um software que gera objetos simuladores de *voxels* mediante imagens de Tomografia Computadorizada (CT). Sua importância para os Sistemas de Planejamento é fundamental, pois permite simular o paciente de maneira mais fidedigna do que com a utilização dos objetos simuladores padrões.

Nesta seção será apresentada uma possível integração entre o IBMC[5] e o *ICCT* a fim de gerar o objeto simulador de *voxels*. As imagens são referentes ao objeto simulador *Rando* da empresa *The Phantom Laboratory Phantoms* (81). Ele foi adquirido pelo Hospital das Clínicas da Faculdade de Medicina da Universidade de São Paulo (HCFMUSP), o qual cedeu algumas imagens desse objeto simulador para o Centro de Engenharia Nuclear - IPEN/CNEN.

O *ICCT* gera quatro arquivos de resultados:

1. *TissueMap.txt*: Contêm matrizes que representam os diferentes tecidos do paciente discretizados por *voxel*. Por questões de modelagem, os valores zeros nas matrizes correspondem aos tecidos de número vinte e quatro.

2. *DensityMap.txt*: Contêm matrizes que representam as densidades dos tecidos discretizadas por *voxel*.

3. *Components.txt*: Contêm a composição dos tecidos no formato do *MCNP*.

4. *Table_ MDId.txt*: Contêm uma tabela com três valores, sendo o primeiro o nome do tecido o segundo a densidade desse material e o terceiro o número de identificação desse material nas matrizes de tecidos.

A seguir foram descritos os passos para criar o objeto simulador de *voxels* mediante o IBMC.

Para auxiliar na extração das informações geradas pelo *ICCT* foi utilizada as classes *GeometryDataMCNP* e *RepeatedStructureForPhantom*. O que mostra mais um vez a reutilização de códigos proporcionada pelo IBMC .

[5] <https://bitbucket.org/fmassicano/examples-of-ibmc/src/0eebe42b4072035f6ec4d58fd99b899a7085f520/src/main/java/examples/icct/?at=master>

Alguns métodos tiveram que ser acrescentados na classe *GeometryDataMCNP* a fim de se adaptar as saídas do *ICCT*. A Figura 5.34 mostra um diagrama de classes de *GeometryDataMCNP* com os métodos utilizados nesse caso. Os métodos novos foram destacados em verde.

O método *setMapTissueDensity* já existia, porém foi criado um novo método[6] com o mesmo nome e parâmetro diferente. Ele recebe o caminho do arquivo *Table_ MDId.txt*. Ele permite a extração das informações do arquivo *Table_ MDId.txt*. Esse método lê esse arquivo e preenche a variável *tissueDensityMap* que é um mapa ***chave*** e ***valor***. A ***chave*** é o número do tecido e o ***valor*** é sua densidade.

O método *setMaterialWithFormatMCMP* extrai as informações dos materiais contidas no arquivo *Components.txt*. Esse arquivo já contêm os materiais escritos no formato do *MCNP*. Dessa forma, esse método somente armazena esse arquivo numa variável a fim dela ser utilizada no Módulo de Construção.

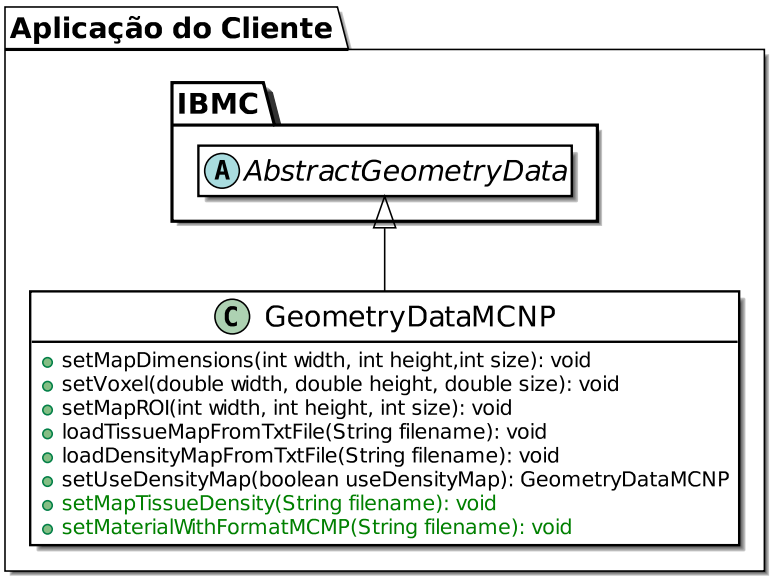

Figura 5.34: Diagrama de classes de *GeometryDataMCNP* somente com os métodos utilizados para o caso da integração do IBMC com o ICCT.

A seguir, na Figura 5.35, é mostrado o diagrama de sequência dos passos que foram feitos para carregar o objeto ***geo*** da classe *GeometryDataMCNP* com os valores dos resultados do *ICCT*.

Primeiramente é criado o objeto ***geo***, da classe *GeometryDataMCNP*, e preenchido com as dimensões das matrizes, contidas no arquivo *config.txt*, com o auxílio dos métodos *set-*

[6] Sobrecarga do método *setMapTissueDensity*.

MapDimensions e *setMapROI*. Depois é carregada a matriz de tecidos através do método *loadTissueMapFromTxtFile*. Os materiais e a tabela de densidade são carregados respectivamente por *setMaterialWithFormatMCMP* e *setMapTissueDensity*. As dimensões dos *voxels* foram fixadas à $0.4cm$ de largura, comprimento e altura.

Por fim foi utilizado o método *setUseDensityMap* com valor "*false*" indicando o uso do mapa carregado pelo método *setMapTissueDensity*. Embora o *ICCT* forneça as matrizes referente à densidade, elas não são utilizadas pois correspondem aos mesmos valores encontrados em *Table_ MDId.txt*.

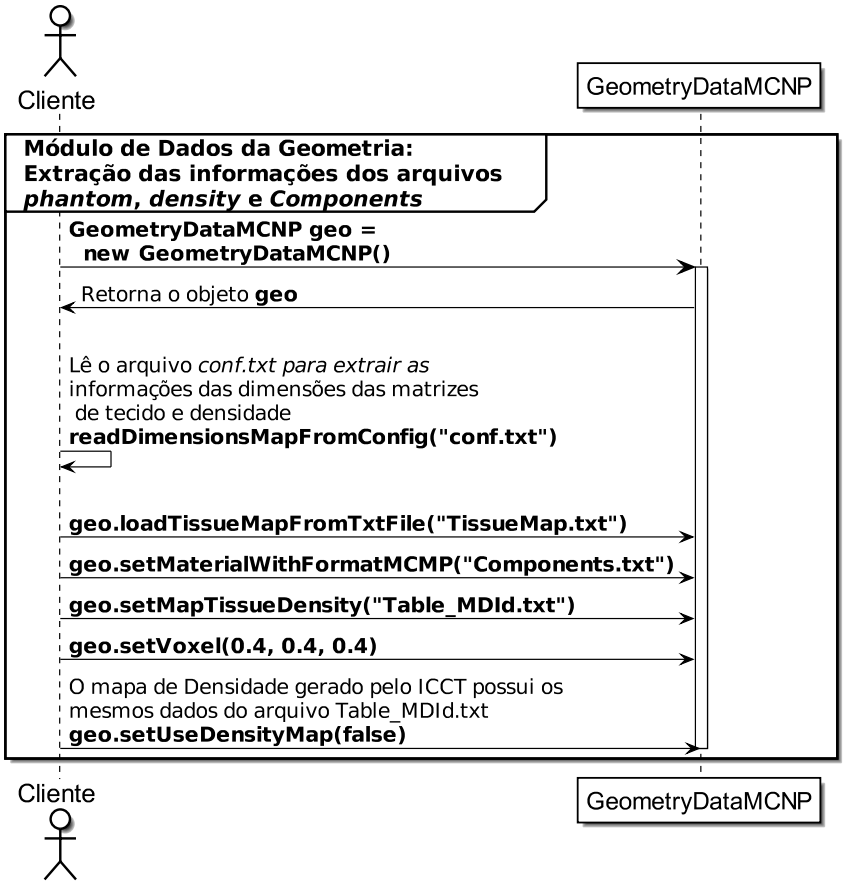

Figura 5.35: Diagrama de sequência referente preenchimento do objeto ***geo*** da classe *GeometryDataMCNP* mediante os dados do *ICCT*.

A Figura 5.36, mostra um diagrama de sequência que apresenta as implementações de alguns métodos com relação à classe *Icct* correspondente ao Módulo de Construção.

No método *cellBlock*, na construção do objeto *rsfphantom*, tem-se que o último parâmetro do construtor recebe um valor *null*. Isto indica que o objeto *rsfphantom* não conterá o objeto *GroupOfMaterials* de *helperMCNP*. Isto ocorre, por que o *ICCT* já fornece o arquivo de material formatado para o *MCNP*. Neste mesmo método os valores de importância são

fixados para fótons e elétrons.

Em *surfaceBlock* somente foi requisitado o método *getSurfaceBlockRS* do objeto *rsfphantom* a fim de extrair a representação da superfície.

Em *dataBlock* foi utilizado o método *getMaterialWithFormatMCMP* a fim de extrair a representação dos materiais que foram especificados pelo *ICCT*.

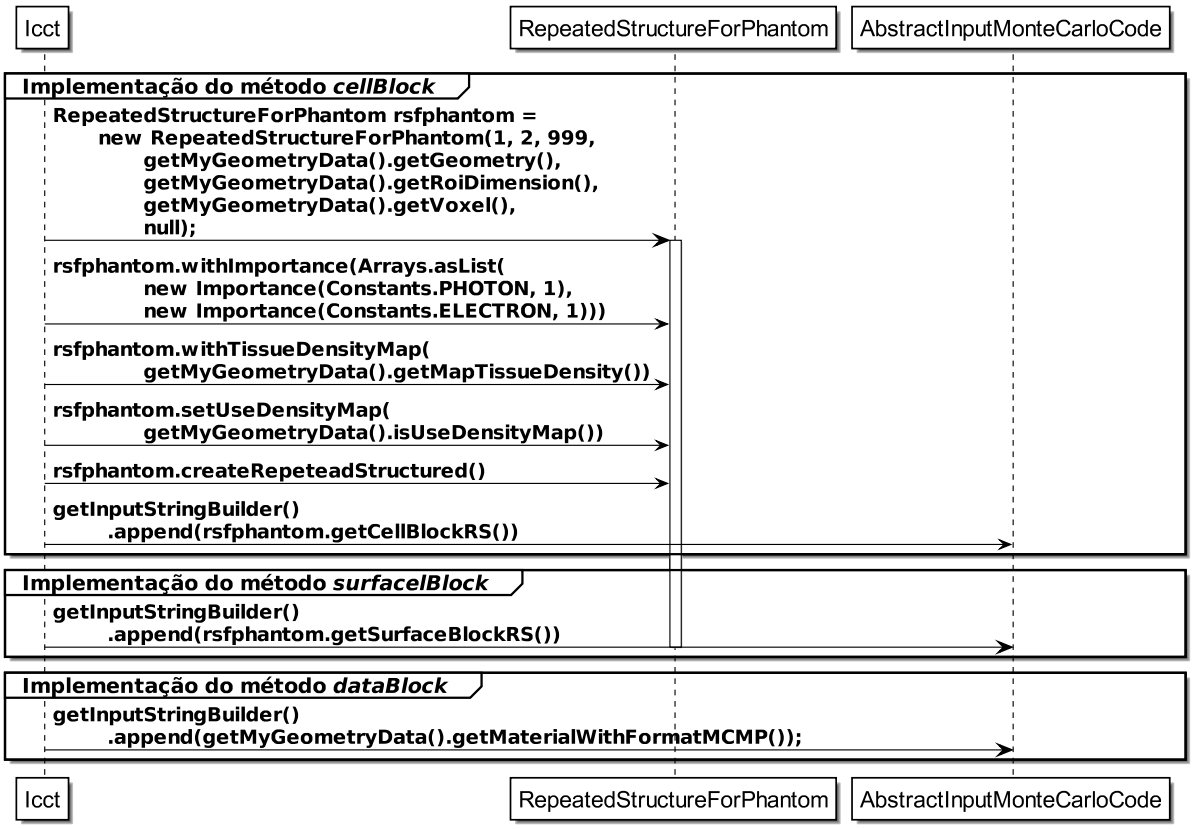

Figura 5.36: Diagrama de sequência referente à criação do arquivo de entrada do MCNP por intermédio da classe *Icct*.

Mediante esses módulos de dados e de construção o objeto simulador de *voxels* foi formado. A Figura 5.37 apresenta uma imagem do objeto simulador carregado no MCNP.

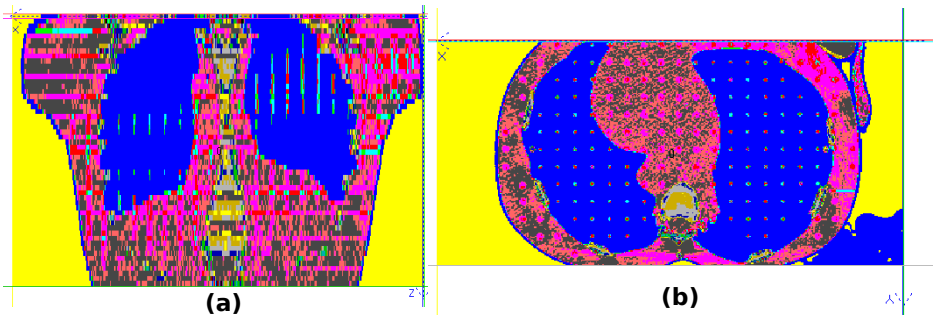

Figura 5.37: Imagem do objeto simulador RANDO® criado no MCNP por intermédio da classe *Icct*.

Um fator importante nessa integração é que ela mostra um aumento no potencial do IBMC, no auxilio da criação de novos arquivos do *MCNP*, a medida que ele é utilizado. Isto porque, muitos códigos já produzidos são reutilizados.

5.2 Casos de uso com o IBMC

Os casos descritos anteriormente foram apresentados com o objetivo de comprovar que o IBMC proporciona um ambiente de desenvolvimento que auxilia na criação de arquivos *INP* complexos.

Nesta seção, foram descritos três casos de uso que possuem a finalidade de apresentar o potencial do IBMC na ajuda aos pesquisadores que lidam com o código *MCNP*.

5.2.1 Primeiro caso de uso: Geração de arquivos complexos do MCNP por usuários iniciantes

Dependendo da complexidade do problema a ser resolvido, principalmente na parte geométrica e na modelagem da fonte de radiação, o processo de construção do arquivo de dados de entrada do código *MCNP* pode se tornar um trabalho dispendioso e complexo.

Na maioria dos casos torna-se necessário a criação de algoritmos que auxiliem na construção destes arquivos, e desta forma, o usuário precisará ter alto conhecimento em desenvolvimento de *software* e do *MCNP*.

Para lidar com este problema, o IBMC pode ser utilizado para auxiliar os usuários iniciantes a produzir códigos complexos, do *MCNP*, em um período razoavelmente curto e sem a necessidade de produzir *softwares* complexos.

Com base nisso, foi elaborado um caso de uso no qual um usuário iniciante pode simular um procedimento de Radioterapia no *MCNP*.

Primeiramente, suponha que já existam os Módulos de Dados e de Construção para um determinado caso de Radioterapia feitos por um outro usuário experiente no IBMC. Suponha, também, que estes Módulos de Dados foram feitos pensando em *softwares* terceiros já conhecidos por esse usuário experiente. Por exemplo, ele desenvolveu um módulo de dados para extrair informações provenientes de um *software* utilizado em sistemas de planejamento em Radioterapia.

Outra suposição neste caso é que o usuário inexperiente não conhece os *softwares* utilizados em rotina clínica para Radioterapia.

Com base nestas informações, será explicado como o usuário inexperiente poderá se utilizar do IBMC a fim de conseguir efetuar suas simulações.

A Figura 5.38 apresenta um esquema do funcionamento do IBMC feito pelo usuário experiente. Mediante este esquema, o primeiro passo do aluno iniciante deve ser estudar como os Módulos de Dados ***Geo*** e ***Fonte*** são preenchidos. Isso é uma tarefa relativamente simples caso ele tenha conhecimento de orientação a objetos.

No segundo passo, o usuário iniciante, faz um *Script* Java que consiga preencher os módulos ***Geo*** e ***Fonte*** com as informações de seu próprio problema. Esta tarefa é um pouco mais complicada que a anterior, porém, é totalmente factível para um iniciante em Java.

Por fim, no terceiro passo o aluno deverá se utilizar da classe *SimulationFileManager* e passar como parâmetro os objetos ***Radioterapia***, ***Geo*** e ***Fonte***.

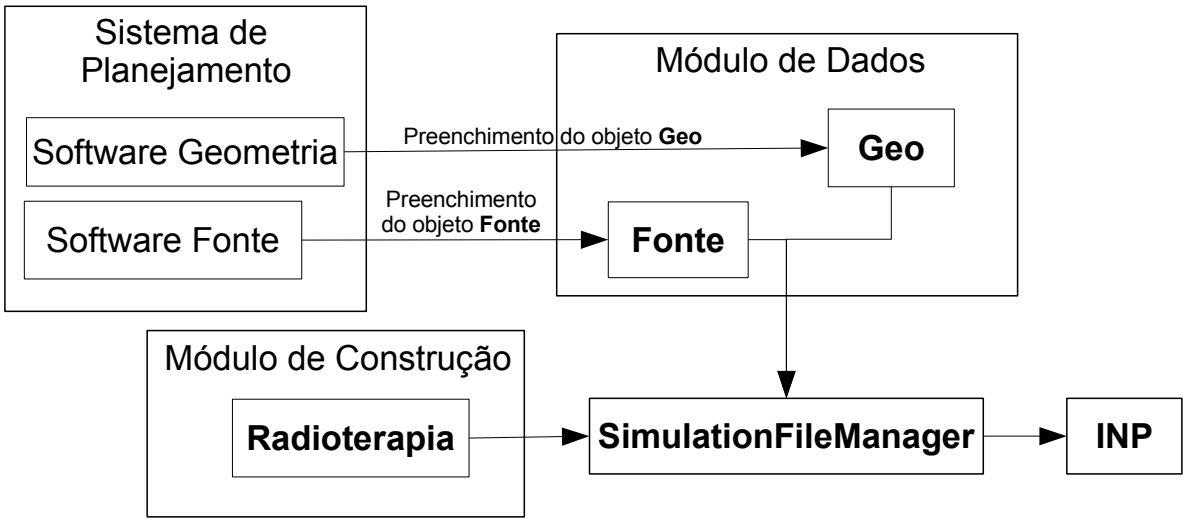

Figura 5.38: Esquema do IBMC feito pelo usuário experiente.

Dessa forma o usuário inexperiente consegue se utilizar daquilo que já foi desenvolvido para assim dar continuidade a sua pesquisa.

5.2.2 *Segundo caso de uso: Adaptar um sistema baseado no IBMC a medida que surgem novas tecnologias*

Outra situação em que o IBMC teria uma contribuição significativa é quando há a inclusão de novas tecnologias nas clínicas médicas a fim de melhorar a dosimetria.

Suponha, em um primeiro momento, um caso hipotético, dentro de uma clínica médica, em que o IBMC foi utilizado com o objetivo de permitir a simulação de casos de Medicina Nuclear. Considere que, os *softwares* responsáveis por adquirirem as imagens provenientes de *PET* e *CT* forneçam essas informações em um formato específico.

Dessa forma, o usuário primeiramente desenvolveu a classe do Módulo de Dados da geometria para interpretar os dados provenientes do *CT*. Depois, ele desenvolveu a classe do Módulo de Dados da fonte de radiação para extrair as informações da fonte de radiação provenientes do *PET*. Com essas duas classes o usuário então criou a classe do Módulo de Construção para gerar o arquivo do *INP* do *MCNP*.

Em um segundo momento, suponha que a clínica adquiriu um novo equipamento de *PET* de outra fabricante que fornece os dados de maneira diferente do equipamento anterior.

Nessa situação, para que o usuário consiga utilizar o *software* desenvolvido com base no IBMC é necessário apenas que ele adapte a classe do Módulo de Dados da fonte de radiação com a saída no novo equipamento. Pode ser que ele precise fazer ajustes na classe do Módulo de Construção.

Caso o usuário ainda não saiba desenvolver essas funcionalidades no IBMC ainda há a opção dele processar os dados do equipamento novo e transformá-lo em dados compatíveis com o antigo. Ele poderia desenvolver isso em qualquer linguagem de programação. Com isso feito, ele já consegue utilizar o *software* desenvolvido com o IBMC para gerar o arquivo *INP* do *MCNP*.

5.2.3 *Terceiro caso de uso: Usar um sistema baseado no IBMC para comparar diferentes métodos*

Considere uma situação hipotética com um sistema baseado no IBMC. Para formar a geometria do problema, o Módulo de Dados da geometria, faz um prévio processo de segmentação nas imagens de *CT*. Com base nisso, um usuário quer modificar esse processo

de segmentação a fim de verificar se essa mudança é significativa no cálculo de dose.

Nessa situação o usuário somente precisa criar um novo método de segmentação na classe correspondente ao Módulo de Dados da geometria. Para isso ele somente precisa entender quais parâmetros o método de segmentação, já implementado, recebe e qual o tipo de dado ele fornece. Com base nisso, o usuário desenvolve um método com essas mesmas características porém com implementação diferente. Dessa forma, é possível substituir o método antigo pelo atual e vice versa.

Isso mostra a flexibilidade de se lidar com um linguagem orientada a objetos. Esse processo torna-se interessante pois o usuário somente precisa entender aquilo que ele irá implementar, sem se preocupar com outras partes do código. Além de não precisar da ajuda do desenvolvedor do código anterior.

Capítulo 6

Conclusões

Para o presente trabalho foi proposto desenvolver um Sistema de Planejamento Dosimétrico que utiliza o código *Monte Carlo N-Particle Transport Code (MCNP6)*, a fim de contemplar os procedimentos de radioterapia e medicina nuclear. E também, que o uso desse sistema fosse destinado a área acadêmica.

A dificuldade em se desenvolver tal sistema foi lidar com as diferenças entre os tipos de tratamentos propostos, como por exemplo, em relação à grande discrepância na caracterização da fonte de radiação em cada um deles.

Com base nisso, foi criado o sistema IBMC com o objetivo de auxiliar pesquisadores de todo o mundo a desenvolverem, de maneira rápida e prática, seus próprios sistemas integrados ao código *MCNP*.

O sistema IBMC mostrou-se capaz de reproduzir os dois tipos de tratamentos propostos. Além disso, ele mostrou-se capaz de servir como uma ferramenta que possibilita criar novos sistemas de planejamentos dosimétricos de maneira prática e rápida. Isso porque o IBMC permite utilizar sistemas já existentes para lidar com as etapas 1 e 3, mostradas na Figura 4.1, fazendo somente a ligação entre essas duas etapas.

O fato de não ser necessário criar ferramentas já consolidadas referente às etapas 1 e 3, simplifica e agiliza consideravelmente o desenvolvimento de novos modelos de sistemas de planejamento dosimétrico com o *MCNP*. Um exemplo disso, foi o resultado de braquiterapia, onde o *software AMIGOBrachy*, já consolidado, foi utilizado como suporte para as etapas 1 e 3, possibilitando assim que um sistema baseado no IBMC pudesse construir um arquivo *INP* similar ao do *AMIGOBrachy*. Assim um novo sistema de planejamento poderia ser criado utilizando essas características já preparadas no *AMIGOBrachy* juntamente com o IBMC.

Na seção 3.5 foi discutido sobre sistemas modulares baseados em códigos de Monte Carlo e o entendimento da estrutura interna desses sistemas contribuiram para o desenvolvimento do IBMC. Em relação ao Voxel2MCNP e ao MINERVA o IBMC mostra a vantagem de permitir o desenvolvedor especificar os padrões para os dados de entrada sem ficar preso aos estabelecidos pelo sistema. Já em relação ao GATE o IBMC poderá ser utilizado para formar a sua linguagem de *script*. Istó mostra a capacidade do IBMC não somente construir o arquivo de entrada para o *MCNP* como também qualquer outro arquivo de entrada de códigos de Monte Carlo.

A evolução do IBMC será um processo contínuo, tendo sempre em vista uma maior flexibilidade em se criar os arquivos *INP* do MCNP. Portanto, para estimular outros desenvolvedores a utilizar e contribuir para o avanço da mesma, ela será disponibilizada como código livre.

Desta forma, o sistema IBMC ajudará não somente novos pesquisadores a desenvolverem seus próprios sistemas de planejamento dosimétrico com o código *MCNP*, como aos pacientes que de forma indireta serão beneficiados com a construção de novas ferramentas que os ajudarão na evolução das terapias que serão submetidos.

Apêndice

Apêndice A

Estrutura do arquivo de configuração do problema (*INP*) do MCNP

O arquivo *INP* possui duas maneiras de ser elaborado, *Initiate-Run* e *Continue-Run*, porém neste trabalho só foi utilizado o modo *Initiate-Run*. Ele apresenta a configuração do problema que será simulado, então nele estão contidos dados da geometria, fonte, *tallys* entre outros dados (77).

Na Figura A.1 foi descrito a estrutura do arquivo *INP* no modo *Initiate-Run* (77).

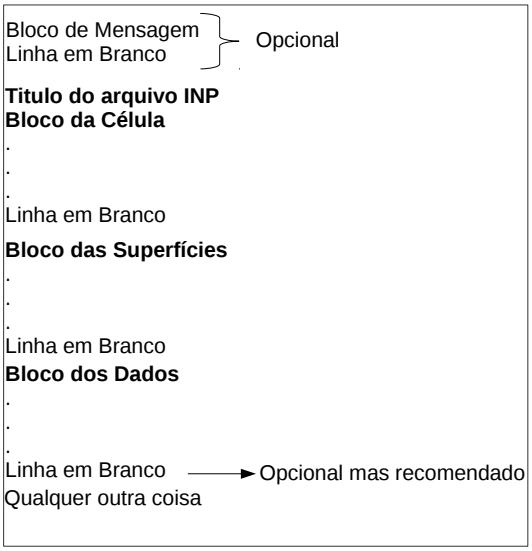

Figura A.1: Estrutura do arquivo de configuração da simulação no MCNP.

Este arquivo é divido por cinco partes mediante às linhas em branco que significam termino dos blocos de informação. O primeiro, bloco da mensagem e o último bloco onde o MCNP não interpreta e serve somente para documentação do problema em questão, são opcionais. Dessa forma, os três principais blocos são o Bloco das Células, o Bloco das Superfícies e os Blocos dos Dados.

Apêndice B

Diagramas de Classes

Nesse trabalho foram apresentados diversos diagramas de classes. Para melhor interpretá-los, encontram-se a seguir as principais representações envolvidas na construção desses diagramas.

Na Figura B.1 é apresentado o diagrama de uma classe concreta. Ela contêm a letra "***C***" dentro de um círculo antes do nome da classe.

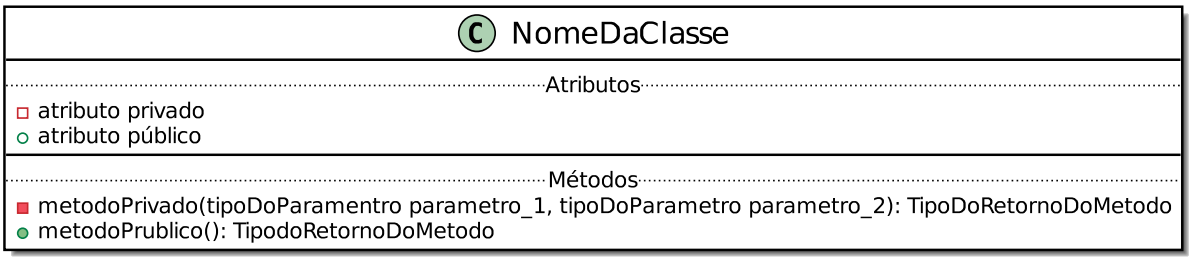

Figura B.1: Representação de uma classe concreta no diagrama de classes.

Na Figura B.2 é apresentado o diagrama de uma classe abstrata. Ela contêm a letra "***A***" dentro de um círculo antes do nome da classe.

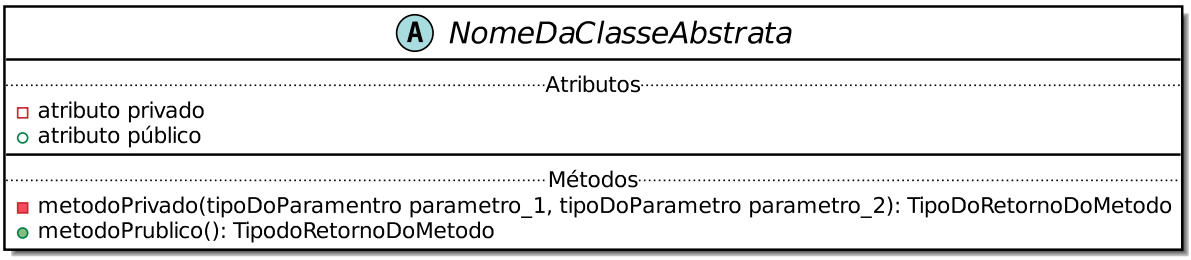

Figura B.2: Representação de uma classe abstrata no diagrama de classes.

Na Figura B.3 é apresentado o diagrama de uma interface. Ela contêm a letra "***I***" dentro de um círculo antes do nome da interface.

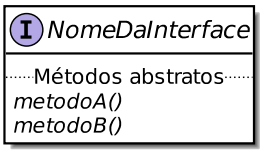

Figura B.3: Representação de uma interface no diagrama de classes.

Na Figura B.4 é apresentada as relações de herança e implementação entre classes.

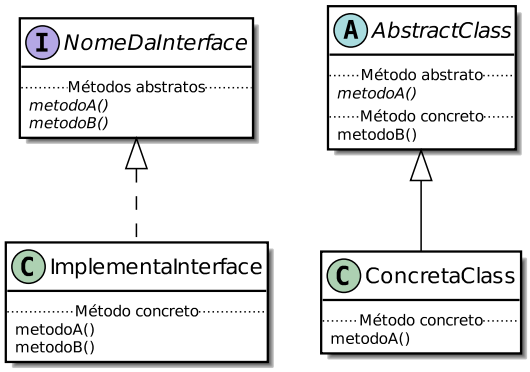

Figura B.4: Representação de relações de herança e implementação no diagrama de Classes.

Na Figura B.5 é apresentada uma relação de agregação e composição entre classes.

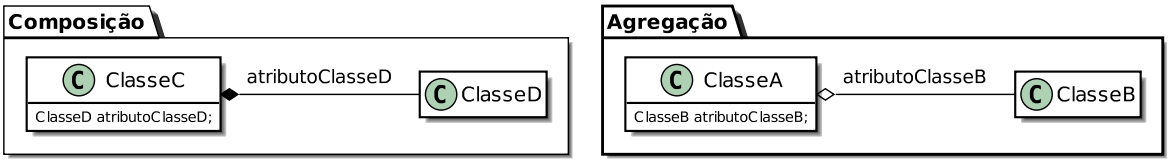

Figura B.5: Representação de relações de agregação e composição no diagrama de Classes.

Apêndice C

Biblioteca *helperMCNP* presente no IBMC.

C.1 Principais características da biblioteca helperMCNP.

A biblioteca *helperMCNP* foi construída para auxiliar na elaboração dos arquivos de entrada para o código *MCNP*. Ainda há algumas funcionalidades do *MCNP* que não são contempladas pela biblioteca. Contudo, a intenção é que ela seja completada continuamente, conforme a necessidade dos desenvolvedores que a utilizarem. A seguir serão descritas todas as funcionalidades presentes nessa biblioteca atualmente.

C.1.1 Construção de Superfícies.

No *MCNP* as superfícies são especificadas mediante equações, pontos ou macrocorpos. Cada superfície divide em dois o espaço, um no sentido positivo e o outro no sentido negativo em relação à superfície. Por intermédio dessas superfícies o MCNP simula a geometria do problema.

Por enquanto a biblioteca *helperMCNP* somente suporta algumas superfícies especificadas por equações e macrocorpos. No entanto, ela pode ser facilmente expandida para suportar as superfícies que faltam, assim como o caso da especificação através de pontos.

Na Figura C.1 é apresentada a classe abstrata *Surface* que servirá de superclasse para todas as classes responsáveis por construir superfícies. As descrições dos principais métodos dessa classe estão na Tabela C.1.

Dessa forma, para se construir um novo tipo de superfície primeiramente é necessário estender a classe abstrata *Surface*. A classe *Surface* implementa a interface *CellGeometry* a qual abstrai todas as maneiras de se caracterizar a geometria de uma célula, sendo que uma delas são as superfícies. Essa classe estende a classe *ParameterInLine* que auxilia na

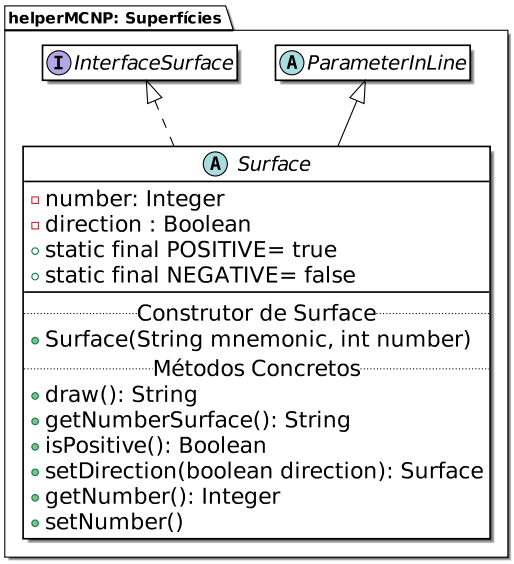

Figura C.1: Diagrama de classes de *Surface*.

construção das superfícies no *MCNP*.

Tabela C.1 - Descrição dos principais métodos da classe *Surface*.

Métodos	Descrições
Construtor de Surface	Recebe dois parâmetros, o mnemônico da superfície que está sendo criada e o número dessa superfície.
draw	Escreve a superfície especificada com a linguagem do *MCNP*.
setDirection	Utilizado quando ocorrer a criação das "células". Recebe como parâmetro um valor booleano que quando verdadeiro indica direção no sentido positivo e quando falso indica direção no sentido negativo. Os valores booleanos podem ser adquiridos pelas variáveis estáticas POSITIVE e NEGATIVE.
isPositive	Retorna verdadeiro se o sentido for especificado como positivo e falso caso contrário.
getNumberSurface	Adquire o número da superfície criada.

A seguir serão descritas as superfícies que estão contidas no *Framework*.

Cilindro: Há diversos modos de se criar cilindros no *MCNP* e eles são apresentados na Tabela C.2. Estes dados foram extraídos do manual do *MCNP* (85).

Tabela C.2 - Parâmetros para a construção de um cilindro no *MCNP*.

Mnemônicos	Descrições	Equação	Entradas do cartão
C/X	Paralelo ao eixo X	$(y-\overline{y})^2+(z-\overline{z})^2-R^2=0$	$\overline{y}$ $\overline{z}$ R

Essa classe é subclasse de *Cylinder* que representa todas as maneiras possíveis de se criar um cilindro no *MCNP*. O construtor da classe *Cx* recebe somente dois parâmetros, o primeiro é o número da superfície e o segundo é o valor do raio do cilindro em relação ao eixo x.

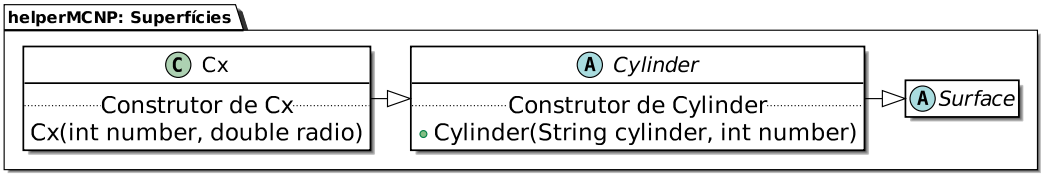

Figura C.2: Diagrama da classe *Cx*.

A seguir, na Figura C.3, há um diagrama de sequência que exemplifica a criação de um objeto da classe *Cx* e como adquirir sua representação para o *MCNP*.

Primeiramente são necessários especificar o número da superfície e o raio do cilindro em relação ao eixo x. Para adquirir a representação no *MCNP* é utilizado o método *draw*.

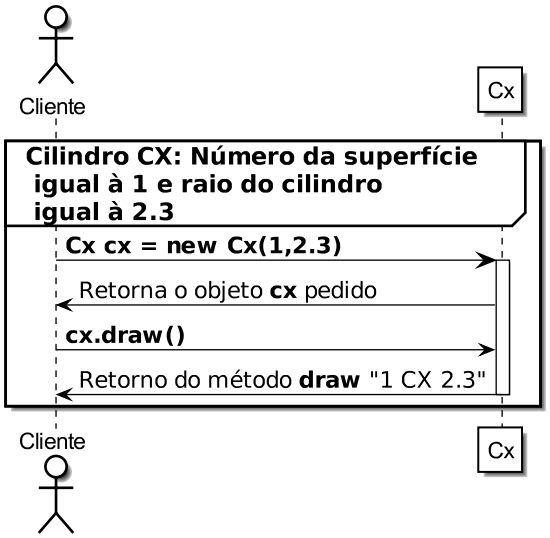

Figura C.3: Diagrama de sequência para exemplificar desde a criação de um objeto da classe *Cx* até o retorno de sua especificação no *MCNP*.

Planos: Os possíveis meios para se especificar um plano no *MCNP* estão decritos na Tabela C.3. Estes dados foram extraídos do manual do *MCNP* (85).

Tabela C.3 - Parâmetros para a construção de um plano no *MCNP*.

Mnemônicos	Descrições	Equação	Entradas do cartão
P	Geral	$Ax + By + Cz - D = 0$	A B C D
PX	Normal ao eixo X	$x - D = 0$	D
PY	Normal ao eixo Y	$y - D = 0$	D
PZ	Normal ao eixo Z	$z - D = 0$	D

Em *helperMCNP*, os planos são definidos pelas classes *Px*, *Py* e *Pz* que criam superfícies planas que interceptam os eixos x, y e z, respectivamente. Isto é, são referentes aos catões ***PX***, ***PY*** e ***PZ***. O diagrama dessas classes estão contidos na Figura C.4.

Todas essas classes são subclasses de *Plane* que representa todas as possíveis formas de planos no *MCNP*. Os construtores de *Px*, *Py* e *Pz* recebem dois parâmetros, o primeiro é o número da superfície e o segundo é o valor da coordenada onde o plano interceptará o eixo x, y ou z, respectivamente.

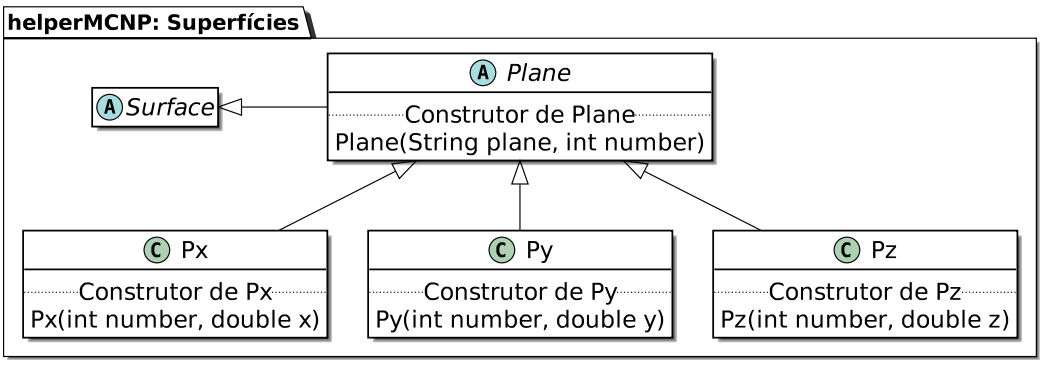

Figura C.4: Diagrama das classes *Px*, *Py* e *Pz*.

A seguir, na Figura C.5, há um diagrama de sequência que exemplifica o uso dessas classes e o modo como devem ser feitas as requisições para obter a representação desses planos para o *MCNP*.

O cliente cria os objetos passando como parâmetro o número da superfície e o valor de suas respectivas coordenadas. Para adquirir a representação dos planos é necessário requisitar o método *draw*.

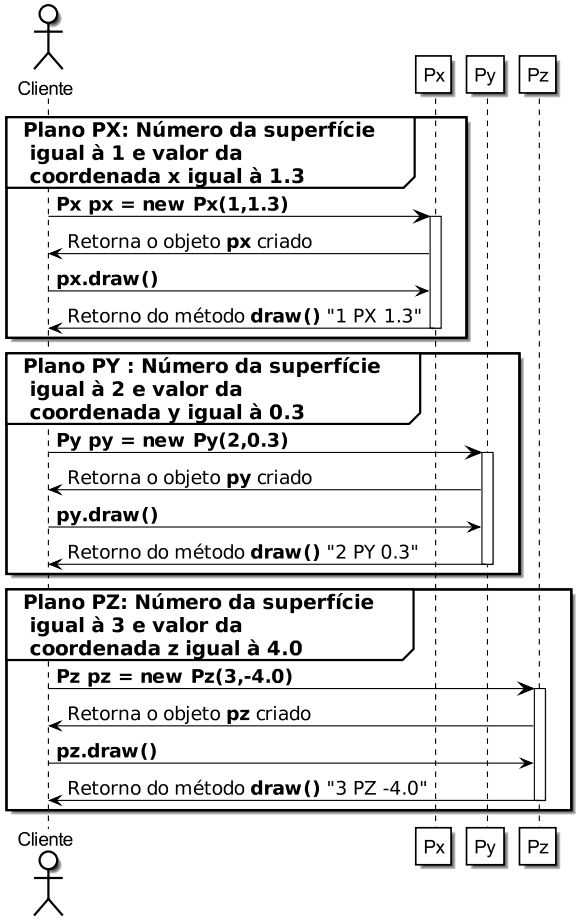

Figura C.5: Diagrama de sequência que exemplifica a criação de objetos de *Px*, *Py* e *Pz* e a extração de suas respectivas saídas para o *MCNP*.

Esferas: Na Tabela C.4 é mostrado os possíveis tipos de construções de uma esfera no *MCNP*. Estes dados foram extraídos do manual do *MCNP* (85).

Tabela C.4 - Parâmetros para a construção de uma esfera no *MCNP*.

Mnemônicos	Descrições	Equação	Entradas do cartão
SO	Centrada na Origem	$x^2 + y^2 + z^2 - R^2 = 0$	R
S	Geral	$(x - \overline{x})^2 + (y - \overline{y})^2 + (z - \overline{z})^2 - R = 0$	$\overline{x}$ $\overline{y}$ $\overline{z}$ R
SX	Centralizado no eixo X	$(x - \overline{x})^2 + y^2 + z^2 - R = 0$	$\overline{x}$ R
SY	Centralizado no eixo Z	$x^2 + (y - \overline{y})^2 + z^2 - R = 0$	$\overline{y}$ R
SZ	Centralizado no eixo Z	$x^2 + y^2 + (z - \overline{z})^2 - R = 0$	$\overline{z}$ R

Há somente um tipo de esfera definida em *helperMCNP*. É a esfera centralizada na origem, declarada pelo cartão ***SO***. Na biblioteca ela é especificada pela classe *So* e seu diagrama de classe é apresentado na Figura C.6.

Essa classe é subclasse de *Sphere* que representa todas as possíveis esferas no *MCNP*. O construtor de *So* recebe somente dois parâmetros, o primeiro é o número da superfície e o segundo é o valor do raio da esfera. Essa esfera estará centralizada na origem.

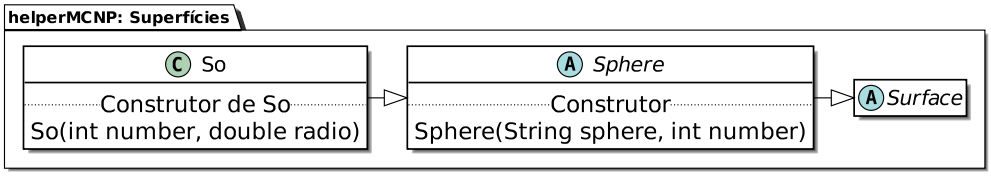

Figura C.6: Diagrama de classes de *So*.

A seguir, na Figura C.7, é apresentado um diagrama de sequência do processo de criação de um objeto de *So*.

Na criação desse objeto é passado o número da superfície e o valor do raio da esfera. Para requerer a representação dessa esfera, no *MCNP*, é utilizado o método *draw*.

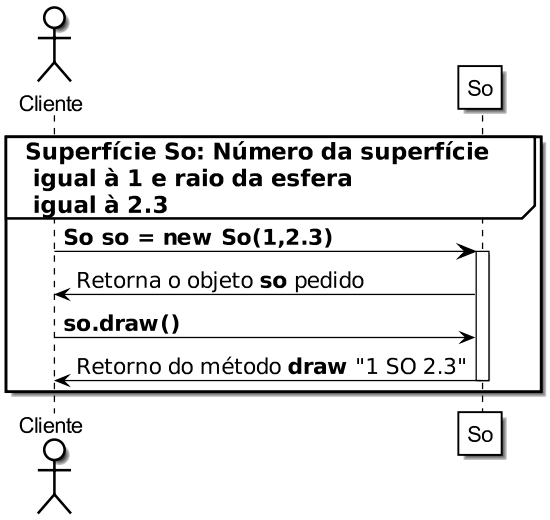

Figura C.7: Diagrama de sequência caracterizando a criação do objeto de *So* e a requisição de sua representação para o *MCNP*.

Cone: Na Tabela C.5 são mostradas todas as possíveis maneiras de se criarem os cones no *MCNP*. Os dados foram extraídos do manual do *MCNP* (85).

Tabela C.5 - Parâmetros para a construção de um cone no *MCNP*.

Mnemônicos	Descrições	Equação	Entradas do cartão
K/X	Paralelo ao eixo X	$\sqrt{(y-\overline{y})^2+(z-\overline{z})^2}-t(x-\overline{x})=0$	$\overline{x}\ \overline{y}\ \overline{z}\ t^2\ \pm 1$
K/Y	Paralelo ao eixo Y	$\sqrt{(x-\overline{x})^2+(z-\overline{z})^2}-t(y-\overline{y})=0$	$\overline{x}\ \overline{y}\ \overline{z}\ t^2\ \pm 1$

Continuação da Tabela C.5

Mnemônicos	Descrições	Equação	Entradas do cartão
K/Z	Paralelo ao eixo Z	$\sqrt{(x-\overline{x})^2+(y-\overline{y})^2}-t(z-\overline{z})=0$	$\overline{x}\ \overline{y}\ \overline{z}\ t^2\ \pm1$
KX	Sobre o eixo X	$\sqrt{y^2+z^2}-t(x-\overline{x})=0$	$\overline{x}\ t^2\ \pm1$
KY	Sobre o eixo Y	$\sqrt{x^2+z^2}-t(y-\overline{y})=0$	$\overline{y}\ t^2\ \pm1$
KZ	Sobre o eixo Z	$\sqrt{x^2+y^2}-t(z-\overline{z})=0$	$\overline{z}\ t^2\ \pm1$

O *helperMCNP* apresenta somente um tipo de cone, o definido sobre o eixo x, especificado pelo cartão ***KX***. Na biblioteca *helperMCNP* ele é especificado pela classe *Kx* e seu diagrama de classe é apresentado na Figura C.8.

Essa classe é subclasse de *Cone* que representa todos os possíveis cones definidos no *MCNP*. Seu construtor recebe somente um parâmetro que é o número da superfície. Os demais parâmetros são especificados pelos métodos *setX*, *setT* e *setSheetCone*.

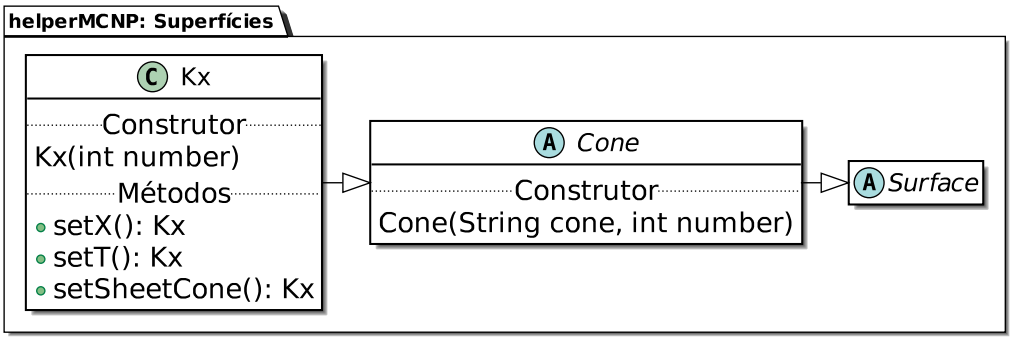

Figura C.8: Diagrama de classes de *Kx*.

A seguir, na Figura C.9, há um diagrama de sequência do processo de criação de um objeto de *Kx* e como ocorre a extração de sua representação para o *MCNP*:

Na criação do objeto é passado somente o número da superfície sendo que os demais parâmetros recorrem aos métodos métodos *setX*, *setT* e *setSheetCone*. Para requerer a representação desse cone, no *MCNP*, é utilizado o método *draw*.

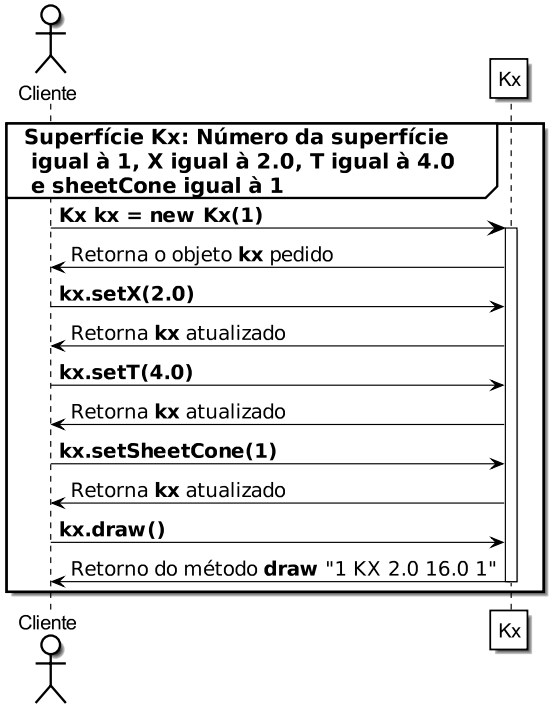

Figura C.9: Diagrama de sequência caracterizando a criação do objeto de *Kx* e a requisição de sua representação para o *MCNP*.

Elipsoide Hiperbólica Paraboloide: A única maneira de se construir uma Elipsoide Hiperbólica Paraboloide no *MCNP* é apresentada na Tabela C.6 sendo seus dados extraídos do manual do *MCNP* (85).

Tabela C.6 - Parâmetros para a construção de uma elipsoide hiperbólica paraboloide no *MCNP*.

Mnemônicos	Descrições	Equação	Entradas do cartão
SQ	Eixo não paralelo à x, y ou z	$A(x-\overline{x})^2 + B(y-\overline{y})^2 + C(z-\overline{z})^2 + 2D(x-\overline{x})+2E(y-\overline{y})+2F(z-\overline{z})+G=0$	A B C D E F G $\overline{x}$ $\overline{y}$ $\overline{z}$

Em *helperMCNP* ele é especificado pela classe *Sq* e seu diagrama de classe é apresentado na Figura C.10.

O construtor recebe somente um parâmetro que é o número da superfície. Os demais parâmetros de *Sq* são especificados pelos diferentes métodos apresentados na classe *Sq*.

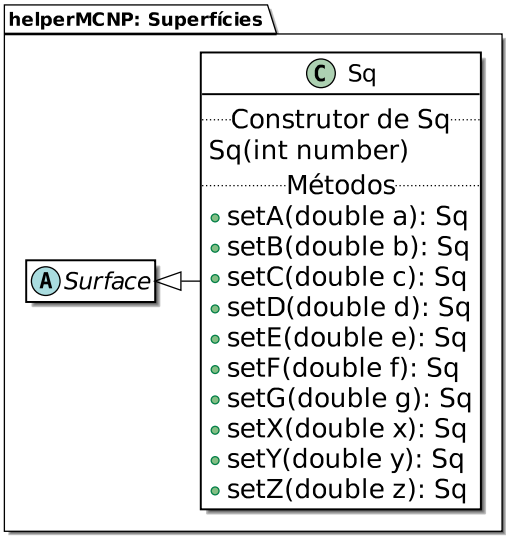

Figura C.10: Diagrama de classes de *Sq*.

A seguir, na Figura C.9, há um diagrama de sequência do processo de criação de um objeto de *Sq* e a extração de sua representação para o *MCNP*.

Na criação do objeto é passado somente o número da superfície já os demais parâmetros são preenchidos pelos métodos *Sets* especificados na classe *Sq*. Para requerer a representação da Elipsoide Hiperbólica Paraboloide, no *MCNP*, é utilizado o método *draw*.

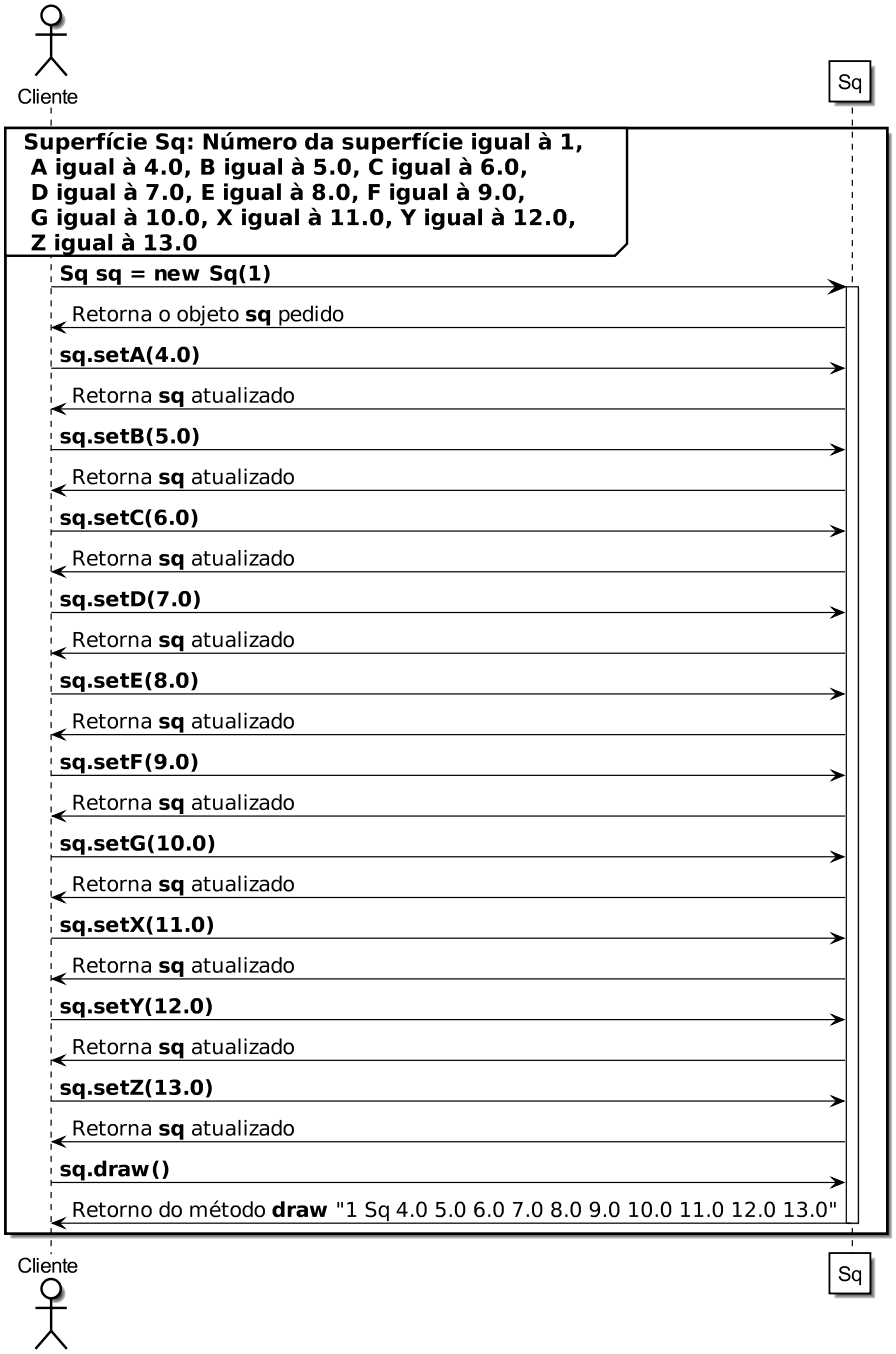

Figura C.11: Diagrama de sequência caracterizando a criação do objeto de *Sq* e a requisição de sua representação para o *MCNP*.

C.1.2 Construção de Células.

Para o *MCNP* uma "célula" é definida como uma região delimitada por superfícies arranjadas entre si mediante intersecções, uniões e complementos. Todas as classes que possuem a finalidade de compor uma "célula" serão descritas a seguir.

No *MCNP* há duas formas de se declarar uma "célula" e ambas foram implementadas na biblioteca *helperMCNP*. A implementação de cada uma delas foi abordada a seguir.

Primeira maneira de se declarar as Células: Essa primeira forma especifica as células conforme o seguinte padrão: "*j m d geom params*". Na Tabela C.7 tem o significado de cada um desses parâmetros. As informações dessa tabela foram retiradas do manual do *MCNP* (85).

Tabela C.7 - Descrição dos parâmetros necessários para se criar uma célula conforme a primeira especificação.

Parâmetros	Descrições
j	Número das células. Possui restrição de $1 \leq j \leq 99.999.999$. Caso a célula seja afetada pelo Cartão TRCL então o limite é modificado para $1 \leq j \leq 99.999$.
m	Número do material. Se *m* igual a zero então a célula é considerada vazia. Possui restrição de $1 \leq m \leq 99.999.999$.
d	Densidade do material. Se negativo a unidade é g/cm^3, mas se for positivo a unidade é $10^{24} atoms/cm^3$. Quando a célula está vazia este parâmetro não deve ser explicitado.
geom	Especifica a geometria da célula através da composição de superfícies mediante o uso dos operadores booleanos UNION, INTERSECTION ou COMPLEMENT.
params	Especificação dos parâmetros de células. Há diversos parâmetros de células: IMP, VOL, PWT, EXT, FCL, WWN, DXC, NONU, PD, TMP, U, TRCL, LAT, FILL, ELPT, COSY, BFLCL, EMBU, e UNC. Estes parâmetros são opcionais e podem ser especificados no bloco de dados do arquivo do MCNP.

Para contemplar a formação da célula segundo a especificação apresentada foi construída a classe *CellSpecification*. Porém a formação do objeto de *CellSpecification* é muito complexo para ser formado por somente uma classe. Na Tabela C.7 tem-se que os três primeiros parâmetros de entrada são especificações simples, com um único valor a ser declarado, e os demais formados por especificações complexas, isto é, precisam de mais informações para serem criados. Dessa forma optou-se pela construção de objetos específicos para as estruturas complexas de forma que elas pudessem ser posteriormente agrupadas à estrutura principal, a fim de compor a célula.

Tendo em vista esta finalidade, foram criadas as interfaces *CellGeometry* e *Parameters*, responsáveis por representar os parâmetros de entrada "*geom*" e "*parameters*" respectivamente, e também a classe abstrata *Complement* responsável por representar um caso particular de *geom*. Essas classes tem a finalidade de auxiliar na construção do objeto de *CellSpecification*. Pode-se dizer que a classe *CellSpecification* é composta por essas classes auxiliares. Na Figura C.12 é apresentado o diagrama de classes de *CellSpecification*.

A classe *CellSpecification* é subclasse de *AbstractCell* que representa as duas formas possíveis de se criar uma célula. A classe *CellSpecification* é composta pelas classes *CellGeometry*, *Parameters* e *Complement*. Para extrair a representação da célula para o *MCNP* é utilizado o método *draw* da classe *CellSpecification*.

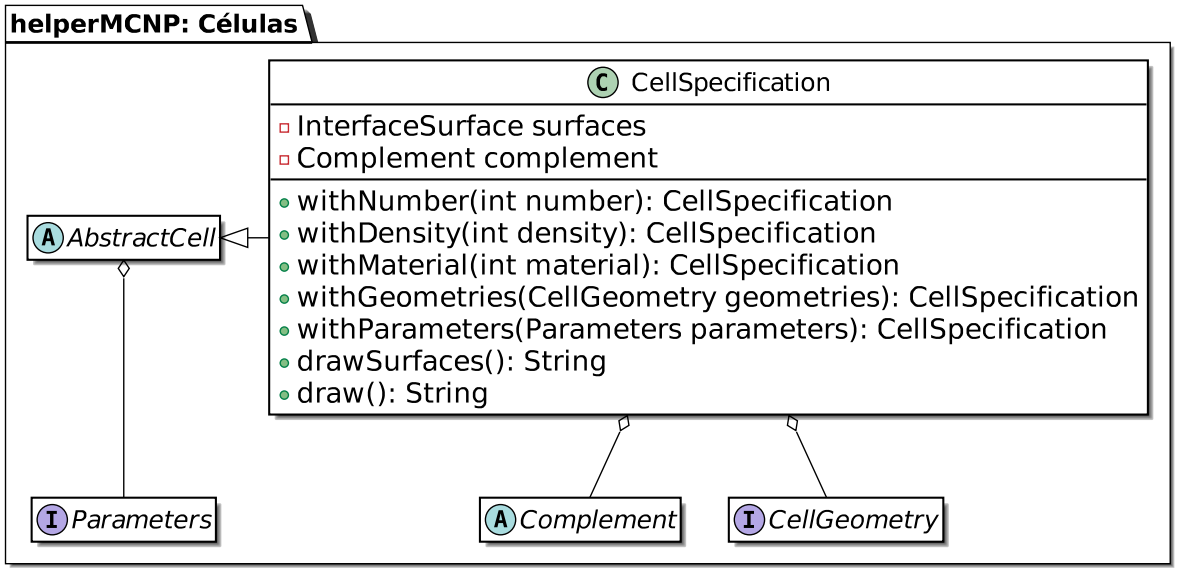

Figura C.12: Diagrama de classes de *CellSpecification*.

A seguir serão descritas as estruturas dessas classes auxiliares começando com as subclasses de *CellGeometry*.

A fim de efetuar a combinação de superfícies, mediante intersecção e união, são utilizadas as subclasses de *GroupSurface* a qual é subclasse de *CellGeometry*. Na Figura C.13 é apresentado o diagrama de classes que mostra a classe *GroupSurface* e suas subclasses.

Ela possui três subclasses principais *Union*, *Intersection* e *ComplexGroupSurface*. Cada uma delas é destinada a um problema particular. A classe *Union* somente realiza a união entre as superfícies, a classe *Intersection* realiza somente a intersecção entre as superfícies e a classe *ComplexGroupSurface* realiza a união ou intersecção entre dois grupos de superfícies ou entre superfícies únicas.

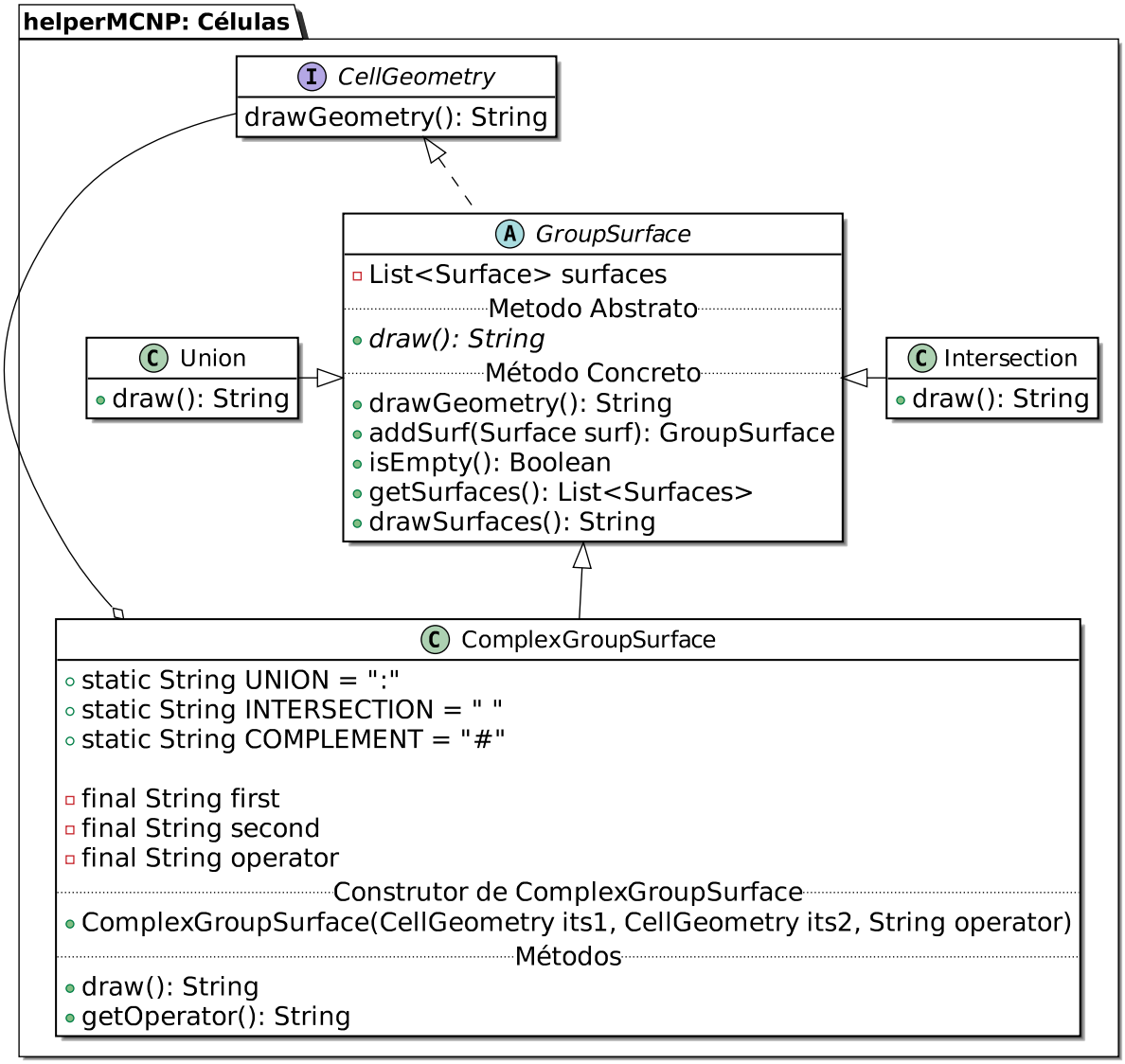

Figura C.13: Diagrama de classes referente à classe *GroupSurface*.

Isto posto, foram elaborados exemplos da utilização de cada uma das subclasses de *GroupSurface*. Primeiramente, foi exemplificado o uso da classe *Intersection* apresentado na Figura C.14.

Primeiramente são criadas as superfícies. Essas superfícies são armazenadas em *Intersection* através do método *addSurf*. É possível especificar o sentido de cada superfície adicionada através do método *setDirection* presente em todas as superfícies. Cada superfície é adicionada como uma intersecção em relação à anterior. Para extrair a representação de *Intersection*, para o *MCNP*, é utilizado o método *draw*.

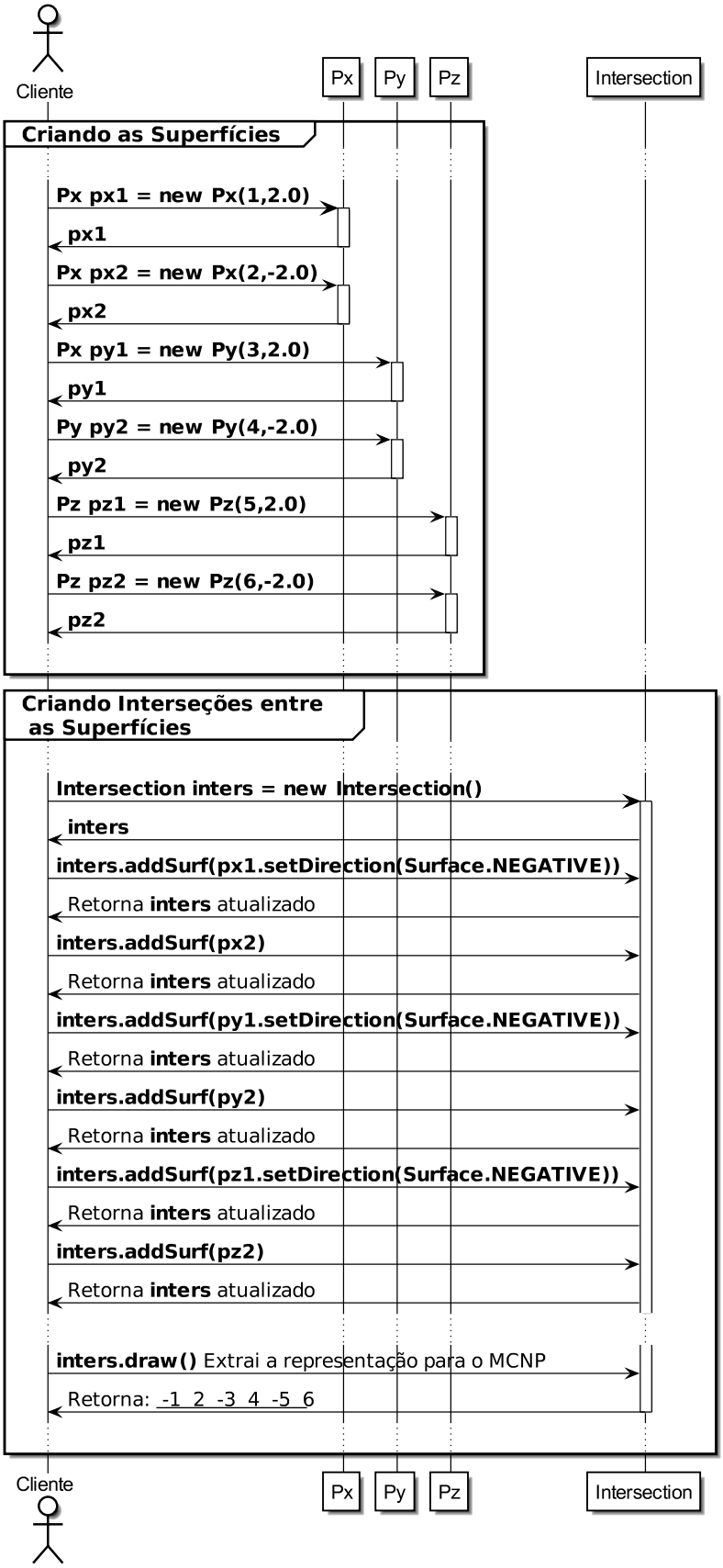

Figura C.14: Diagrama de sequência que exemplifica o uso da classe *Intersection*.

O segundo exemplo, presente na Figura C.15, apresenta o uso da classe *Union*.

Considere que as superfícies *px1*, *px2*, *py1*, *py2*, *pz1* e *pz2* já foram criadas. Essas superfícies são armazenadas em *union* através do método *addSurf*. É possível especificar o sentido de cada superfície adicionada através do método *setDirection* presente em todas as superfícies. Cada superfície é adicionada como uma união em relação à anterior. Para extrair a representação de *Intersection*, para o *MCNP*, é utilizado o método *draw*.

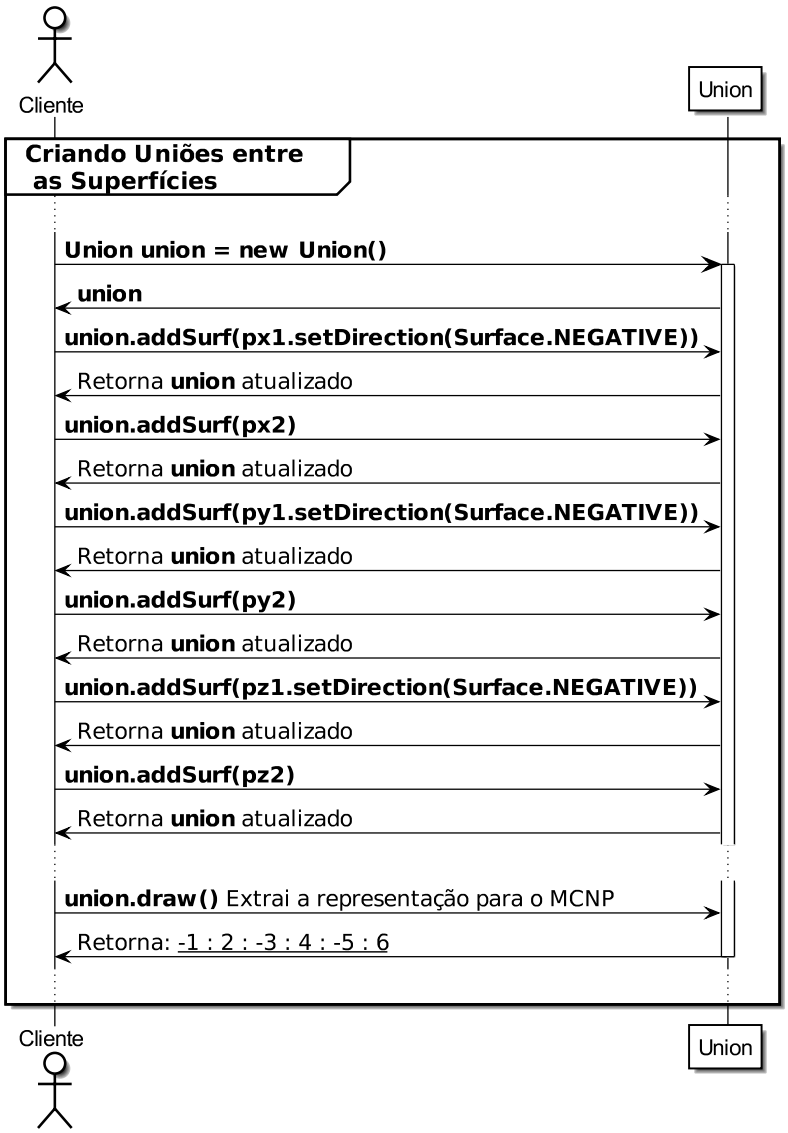

Figura C.15: Diagrama de sequência que exemplifica o uso da classe *Union*.

O último exemplo com relação às subclasses de *GroupSurface* é o uso de *ComplexGroupSurface*. Essa classe mistura combinações que exigem tanto intersecção, união e até mesmo complemento. A seguir, na Figura C.16, há um exemplo do uso dessa classe.

Considere que já foram criados o grupo de intersecção *inter* com representação "1 -2

-3" e o de união *union* com representação "4:5:-6". O construtor dessa classe recebe dois parâmetros da classe *CellGeometry* e o terceiro é uma *String* que representa os operadores Booleanos de Intersecção, União e Complemento que são fornecidos pela própria classe *ComplexGroupSurface* como variáveis estáticas. Esses operadores possuem o papel de combinar os objetos de *CellGeometry*. Para extrair a representação de *ComplexGroupSurface*, para o *MCNP*, é utilizado o método *draw*.

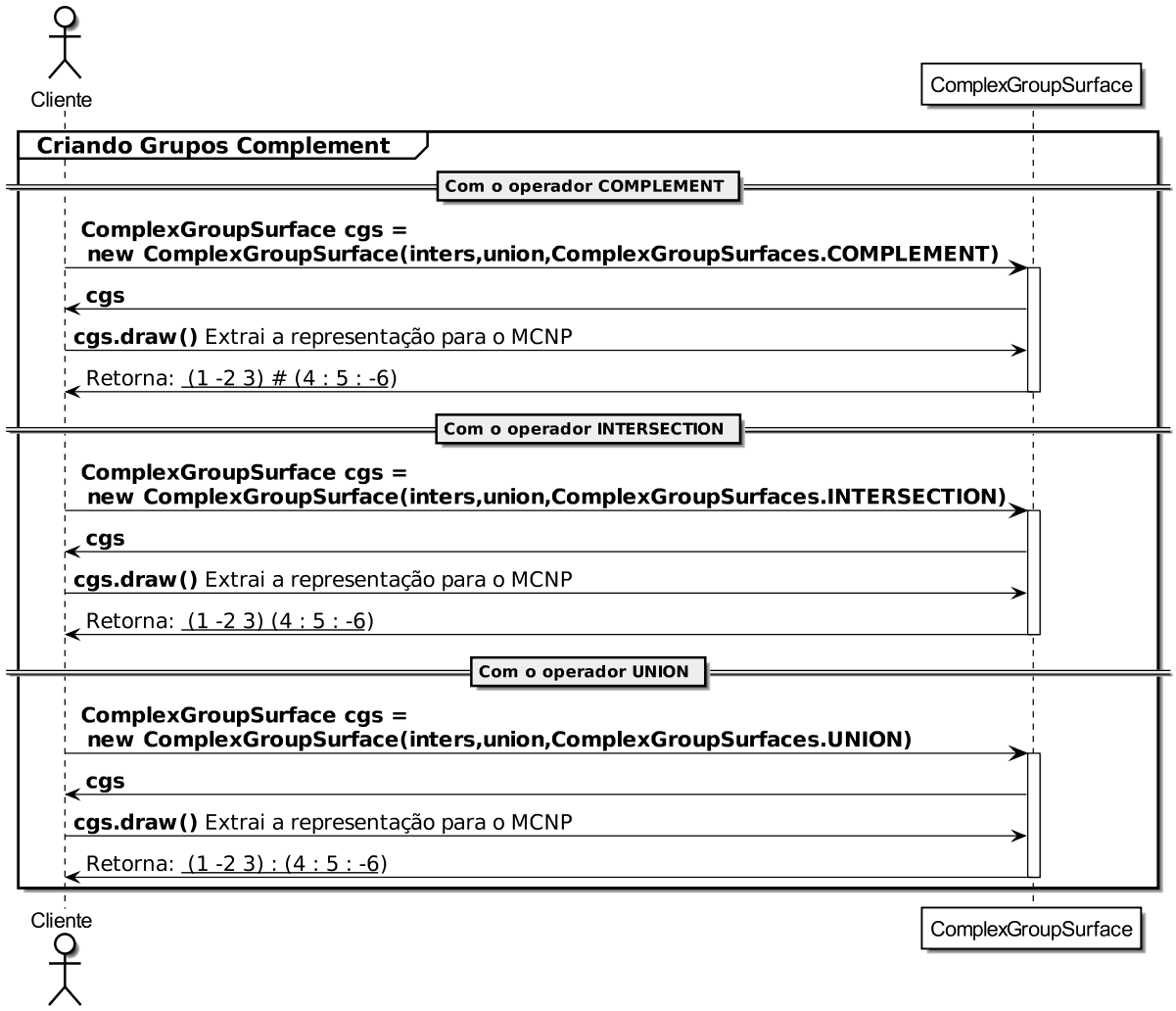

Figura C.16: Diagrama de sequência que exemplifica o uso da classe *ComplexGroupSurface*.

Para acoplar o objeto de *CellGeometry* em *CellSpecification* é utilizado o método *withGeometry*. Nos casos em que se deseja somente considerar uma única superfície na geometria da célula, é somente necessário criar um objeto da classe *Surface* e passá-lo como parâmetro para o método *withGeometries*, indicando o sentido a ser considerado.

Há casos em que uma célula deseja excluir uma célula presente em seu interior, ou excluir uma determinada combinação de superfícies, para isso, a classe *CellSpecification* se

utiliza da classe *Complement*. A seguir é mostrado o diagrama de classes relacionado à *Complement*, apresentado na Figura C.17.

Para que uma célula exclua uma outra célula presente em seu interior é utilizada a subclasse *CellComplement* a qual recebe um lista de células a serem excluídas. No caso em que a célula deseja excluir um grupo de superfícies é utilizada a classe *SurfaceComplement* que recebe um objeto de *CellGeometry* no seu construtor.

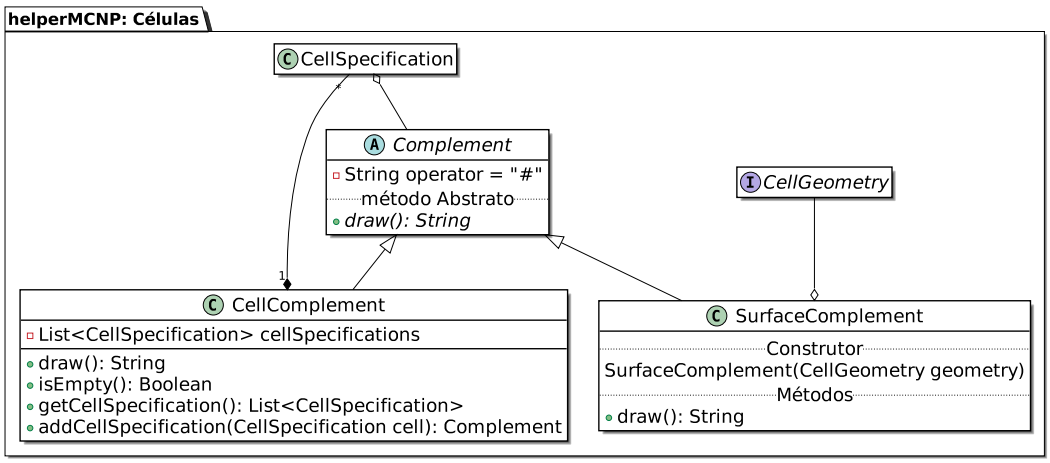

Figura C.17: Diagrama de classes relacionado à *Complement*.

Assim sendo, as duas formas de elaborar o complemento através biblioteca *helperMCNP* serão descritas a seguir. O primeiro caso é quando a célula exclui uma outra que está contida nela. Para isso, é utilizada a classe *CellComplement*. A Figura C.18 mostra um exemplo do uso dessa classe.

Considere que as células *cell_1* e *cell_2* já foram criadas previamente. A classe *CellComplement* recebe um lista de células a serem excluídas. Essas células são adicionadas em *CellComplement* mediante o método *addCellSpecification*. Para requerer sua representação no MCNP é usado o método *draw*.

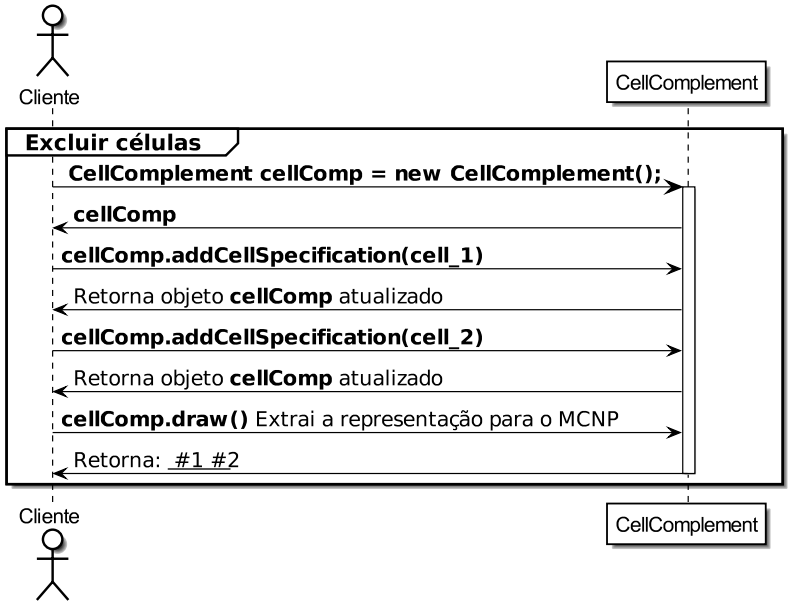

Figura C.18: Diagrama de sequência relacionado à *CellComplement*.

O segundo caso considera a exclusão de uma composição de superfícies ou de uma única superfície. A Figura C.19 mostra um exemplo desse caso.

Considere que as variáveis *inter*, Intersecção de duas superfícies com resultado "-1 2" e *so*, uma esfera com número de identificação igual à 4, já foram criadas previamente. A classe *SurfaceComplement* recebe, através de seu construtor, um objeto de *CellGeometry* que deve ser excluído da geometria da célula que o utilizar. Para requerer sua representação no *MCNP* é usado o método *draw*.

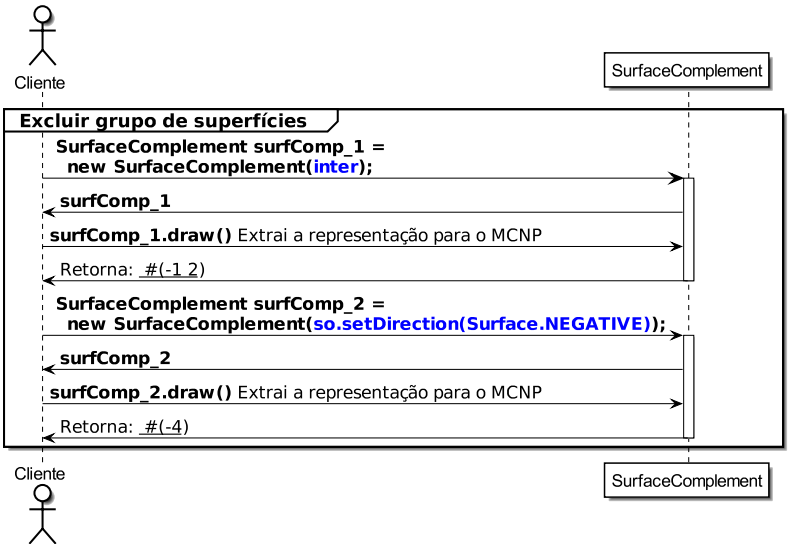

Figura C.19: Diagrama de sequência relacionado à *SurfaceComplement*.

O último parâmetro de entrada a ser considerado são os parâmetros de células listados

na coluna *params* da Tabela C.7. Para lidar com esses parâmetros foi criada a classe abstrata *Parameters* que representa todos os parâmetros possíveis na especificação de uma célula. A Figura C.20 mostra o diagrama de classes concernente aos parâmetros de uma célula.

Somente cinco parâmetros de células foram incluídos na biblioteca *helperMCNP* que são o *Fill*, *FillFull*, *Universe*, *Lattice* e *Importance*. A Classe abstrata *Parameters* possui o método abstrato *draw* responsável por caracterizar a representação de cada parâmetro no *MCNP* e ela também possui o método abstrato *position* o qual especifica uma posição para cada parâmetro de célula, isto porque, uma célula pode ter mais de um parâmetro de célula e assim *position* mantêm a ordem entre eles.

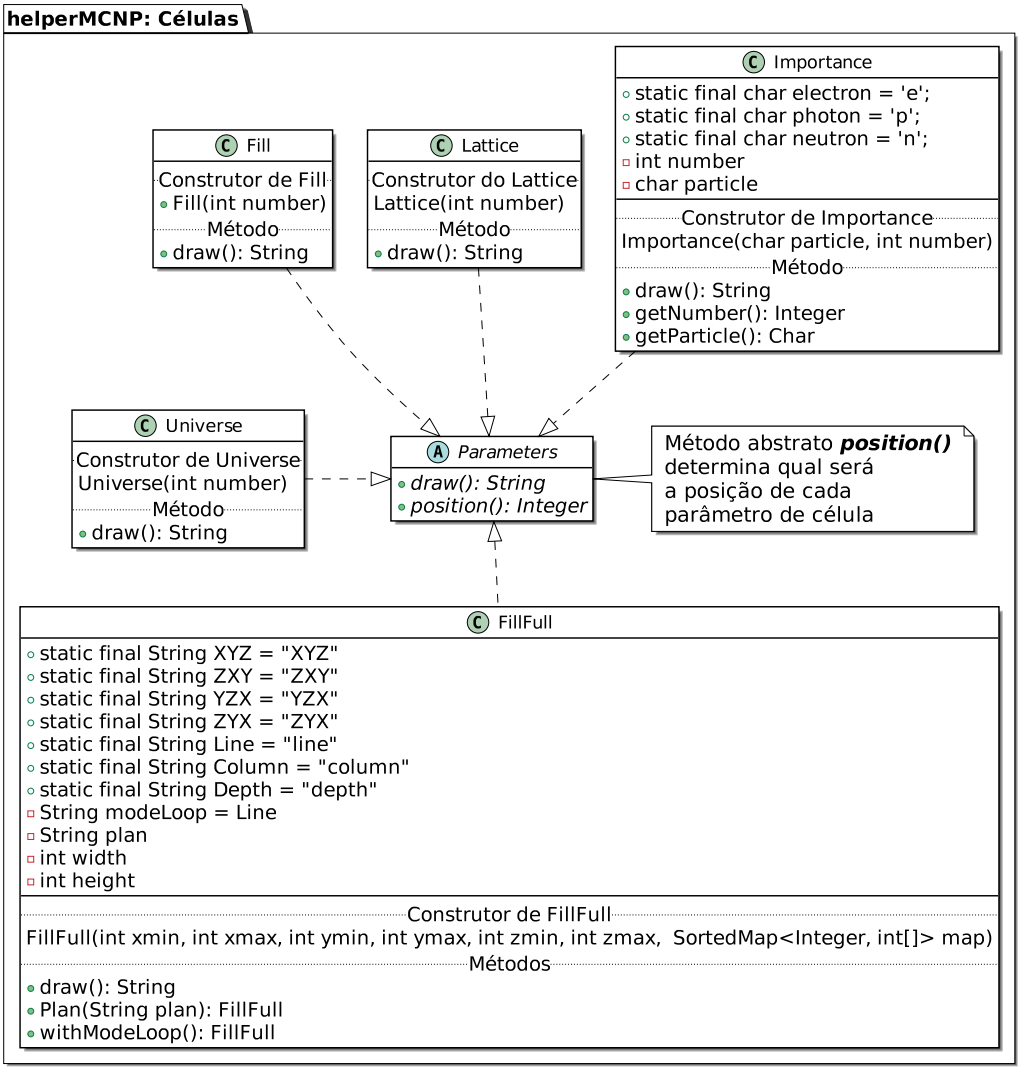

Figura C.20: Diagrama de classes concernente aos possíveis parâmetros de uma célula.

Os quatros primeiros parâmetros que serão descritos a seguir, *Universe*, *Fill*, *FullFill* e

Lattice estão relacionados com o uso de estrutura repetida definida no *MCNP*.

O objetivo da estrutura repetida é possibilitar escrever apenas uma única vez, células ou superfícies que aparecem inúmeras vezes na geometria do problema. Esse tipo de abordagem é muito utilizado, por exemplo, para simular a geometria de um paciente no *MCNP*. Nesse caso a estrutura repetida é utilizada a fim de simular os elementos de volumes ou *voxels* formados por uma sequencia de imagens tomográficas.

O primeiro parâmetro descrito será o *Universe*. Cada célula pode representar um *Universe* e no *MCNP* ele é especificado pelo cartão *U*. A sintaxe desse cartão é: "*U=número*". A biblioteca *helperMCNP* o declara através da classe *Universe*, que recebe em seu construtor um número o qual indica o número de *Universe*. A Figura C.21 apresenta um exemplo da criação de um *Universe* em *helperMCNP*.

A classe *Universe* recebe no construtor o número de *Universe* que a célula representa. Para extrair a representação de *Universe* é utilizado o método *draw*.

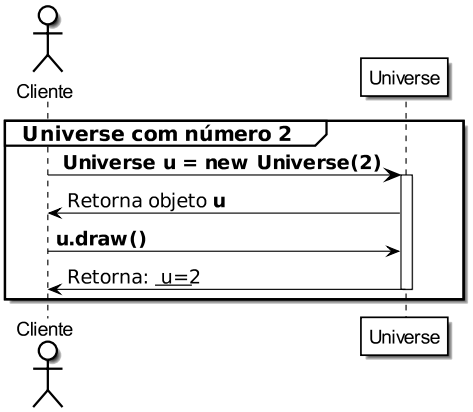

Figura C.21: Diagrama de sequência da criação de *Universe*.

O *MCNP*, através do *Lattice*, possibilita uma célula ser preenchida por inúmeros hexaedros, sólidos com seis faces, ou por inúmeros prismas hexagonais, sólidos com oito faces. A sintaxe desse cartão é "*Lat=número do lattice*", onde o número do *Lattice* vale 1 para hexaedro e 2 para prismas hexagonais.

A biblioteca *helperMCNP* especifica esse parâmetro através da classe *Lattice*. A Figura C.22 mostra um diagrama de sequência que exemplifica o uso dessa classe.

A classe *Lattice* recebe no construtor o número do *Lattice* que pode ser 1, para preencher a célula com hexaedros ou 2 para preencher com prismas hexagonais. Para extrair a representação de *Lattice* é utilizado o método *draw*.

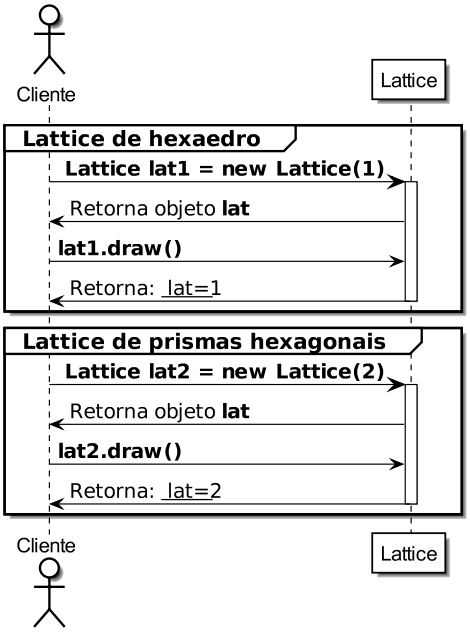

Figura C.22: Diagrama de sequência da criação do *Lattice*.

O próximo parâmetro a ser descrito é o *Fill*. Esse parâmetro permite indicar por qual *Universe* a célula em questão é preenchida. Sua representação no *MCNP* é : "*fill=número de Universe*". Há outra forma de se declarar o parâmetro *Fill*, que será descrita a seguir, mas na forma apresentada aqui, caso a célula seja composta por outras, todas elas serão preenchidas pela mesmo *Universe*. A declaração desse parâmetro por *helperMCNP* é mediante a classe *Fill*. A Figura C.23 apresenta um exemplo de uso dessa classe.

A classe *Fill* recebe no construtor o número do *Universe* que a célula será preenchida. Para extrair a representação de *Fill* é utilizado o método *draw*.

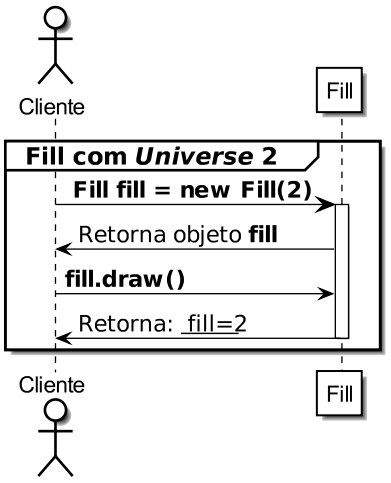

Figura C.23: Diagrama de sequência da criação de *Fill*.

Outra forma de se declarar o parâmetro *Fill* é através da seguinte sintaxe: "fill i_1:i_2 j_1:j_2 k_1:k_2 n_{111} n_{211} ... n_{ijk}".

Ela permite que se uma célula é formada por outras células, as células internas possam ser preenchidas por diferentes *Universe*. Os valores de "i_1:i_2 j_1:j_2" indicam os limites da região a ser considerada pelo *Fill*, e os valores de "n_{111} n_{211} ... n_{ijk}" são os respectivos *Universe* dispostos conforme as direções de "*i, j e k*".

A classe *FillFull* recebe no construtor sete parâmetros, os seis primeiros são referentes à "i_1:i_2 j_1:j_2 k_1:k_2" e o último é um mapa dos *Universe*, sendo fornecido na forma da classe *SortedMap* da biblioteca do Java. Para indicar qual sentido de *i, j e k* é utilizado o método *plan* e para especificar como a matriz 3D deve ser percorrida é utilizado o método *withModeLoop*, suas explicações estão na Figura C.25. Para extrair a representação de *FillFull* é utilizado o método *draw*.

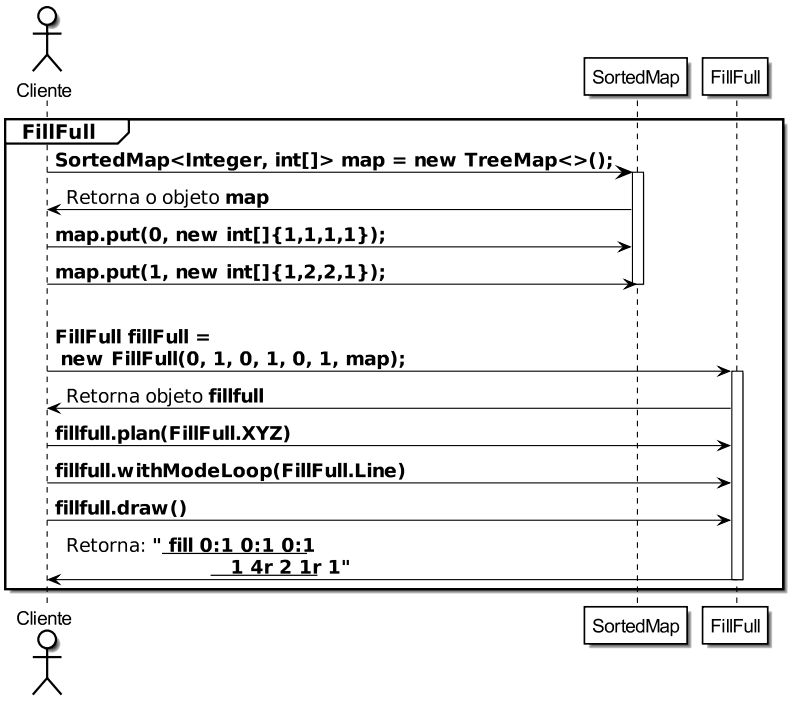

Figura C.24: Diagrama de sequência da criação de *FillFull*.

Ao percorrer esse cubo, a classe *FillFull* considera que o *voxel* inicial sempre será o primeiro do canto superior esquerdo, sendo que a face inicial é a superior. Na Figura C.25 os vetores *i, j e k* estão em vermelho. O método *plan*, indica qual serão as coordenadas em *X, Y, Z* correspondentes à *i, j e k*. O método *withModeLoop* indicará à Classe *FillFull* como os *voxels* devem ser percorridos a fim de extrair os valores de "n_{111} n_{211} ... n_{ijk}". Há somente duas possibilidades de percorrer a matriz de *Universe*, definida pelo método *withModeLoop*.

As duas possibilidades são percorrer em "Linha" ou "Coluna". Considerando primeiramente o sentido i como "Linha" e j como "Coluna". Em "Linha" é percorrido inicialmente o sentido i, depois o j e por fim o k. Em "Coluna" é percorrido inicialmente o sentido j , depois o i e por fim o k. Sendo que os sentidos estão apresentados na Figura C.25.

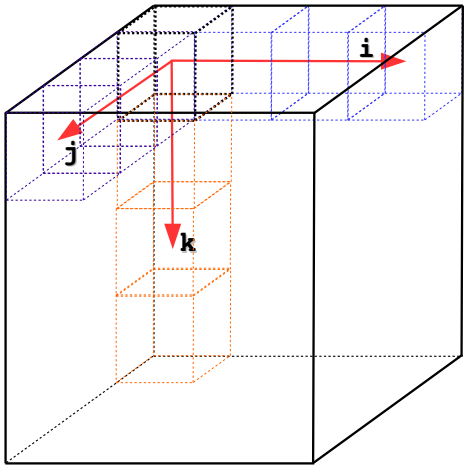

Figura C.25: A imagem apresenta um cubo formado por *voxels*.

Com a finalidade de reduzir o tempo de computação o *MCNP* disponibiliza algumas técnicas de redução de variância. Um dos cartões que o *MCNP* disponibiliza para essa finalidade, e que é que um parâmetro de célula, é a Importância. Sua sintaxe no *MCNP* é "*IMP:p=x*", onde "p" é o símbolo de uma determinada partícula e x é o valor da importância dessa partícula para a célula em questão. A classe *Importance*, em *helperMCNP*, especifica esse cartão do *MCNP*. A seguir, na Figura C.26, encontra-se alguns exemplos de seu uso.

A classe *Importance* recebe no construtor dois parâmetros, o primeiro em uma variável *char* que indica qual partícula será considerada na importância, o segundo é o número da importância de que a partícula terá na célula em questão. No momento, há somente três partículas consideradas, Fótons, Elétrons e Nêutrons, porém futuramente poderão ocorrer inclusão de novas partículas. Para extrair a representação de *Importance* é utilizado o método *draw*.

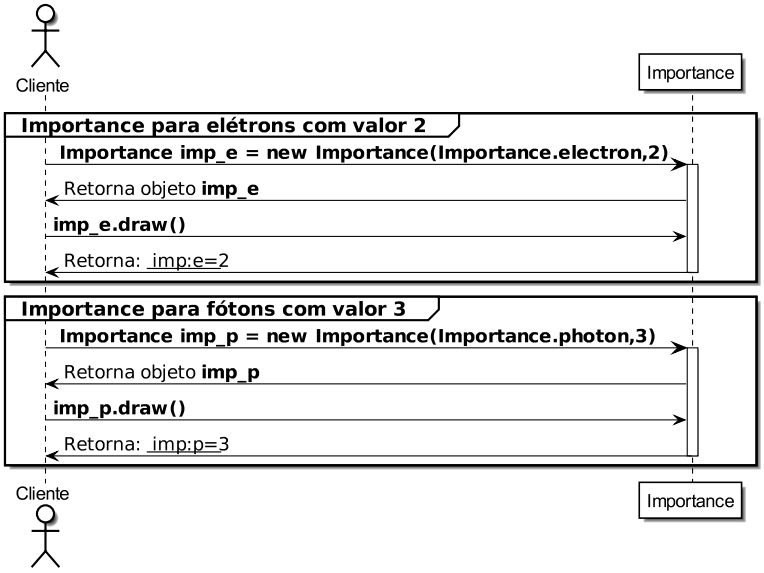

Figura C.26: Diagrama de sequência da criação de *Importance*.

Após apresentar todos os parâmetros de entrada para criação de uma célula no padrão "*j m d geom params*", foram elaborados cinco exemplos, utilizando a biblioteca *helperMCNP*, onde são criadas as células segundo essa sintaxe. Como são somente exemplos, as superfícies que estão sendo criadas não são levadas em consideração em relação a sua posição no plano cartesiano, somente interessando os seus números de identificação.

No primeiro exemplo temos:

```
1  c Exemplo 1
2  1  3  -0.5  -1  2 -3 4  imp:e=1 u=2 lat=1
```

Nesse exemplo está definida uma célula com os seguinte parâmetros de entrada:

- Número de identificação igual à um;
- Número do material igual a três;
- Densidade igual a $0.5 g/cm^3$;
- Geometria formada pela intersecção de superfícies: "-1 2 -3 4";
- Parâmetros de Célula: Importância, *IMP*, para elétrons igual à um, Universo, *U*, igual à dois e Lattice, *LAT*, igual à um;

O diagrama de sequência na Figura C.27 mostra a criação dessa célula pela *helperMCNP*.

A classe *CellSpecification* recebe o número de parâmetros através do método *withNumber*, depois recebe o número do material mediante o método *withMaterial* e a seguir acrescenta a densidade pelo método *withDensity*. A geometria é criada pela classe *Intersection* e depois acoplada à *CellSpecification* pelo método *withGeometry*. Já os parâmetros de células são criados pelas classes *IMP*, *Universe* e *Lattice* sendo por fim acopladas à *CellSpecification* pelo método *withParams*. Para extrair a representação dessa célula no *MCNP* é utilizado o método *draw*.

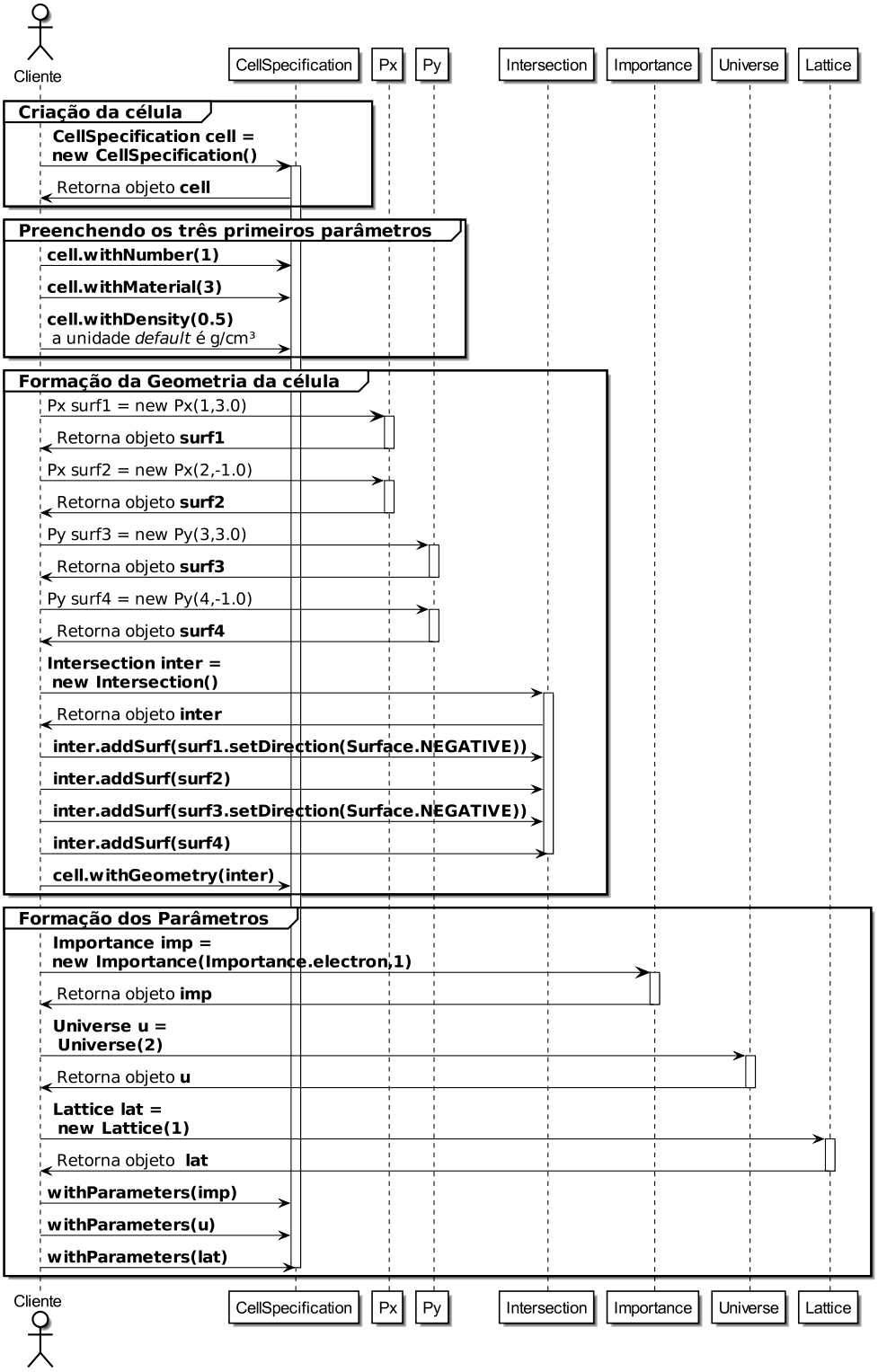

Figura C.27: Diagrama de sequência da criação de uma célula conforme o primeiro exemplo.

O segundo exemplo é mostrado a seguir.

```
1  c celula 2
2  2  4  -1.5  -5 : 6 : -7 : 8  imp:p=1 lat=2
```

Onde os parâmetros de entrada são:

- Número de identificação igual à dois;
- Número do material igual a quatro;
- Densidade igual a $1.5 g/cm^3$;
- Geometria formada pela união de superfícies: "-5 : 6 : -7 : 8";
- Parâmetros de Célula: Importância, *IMP*, para fótons igual à um e *Lattice*, *LAT*, igual à dois;

Na Figura C.28 é mostrado o diagrama de sequência que mostra a criação dessa célula pela *helperMCNP*.

Assim como no primeiro exemplo, Figura C.28, a classe *CellSpecification* recebe os três primeiros parâmetros através dos métodos *withNumber*, *withMaterial* e *withDensity*. A geometria é criada pela classe *Union* e acoplada à *CellSpecification* pelo método *withGeometry*. Os parâmetros de células são criados pelas classes *IMP*, e *Lattice* sendo por fim acopladas à *CellSpecification* pelo método *withParams*. Assim como no primeiro exemplo, para extrair a representação dessa célula no *MCNP* é utilizado o método *draw*.

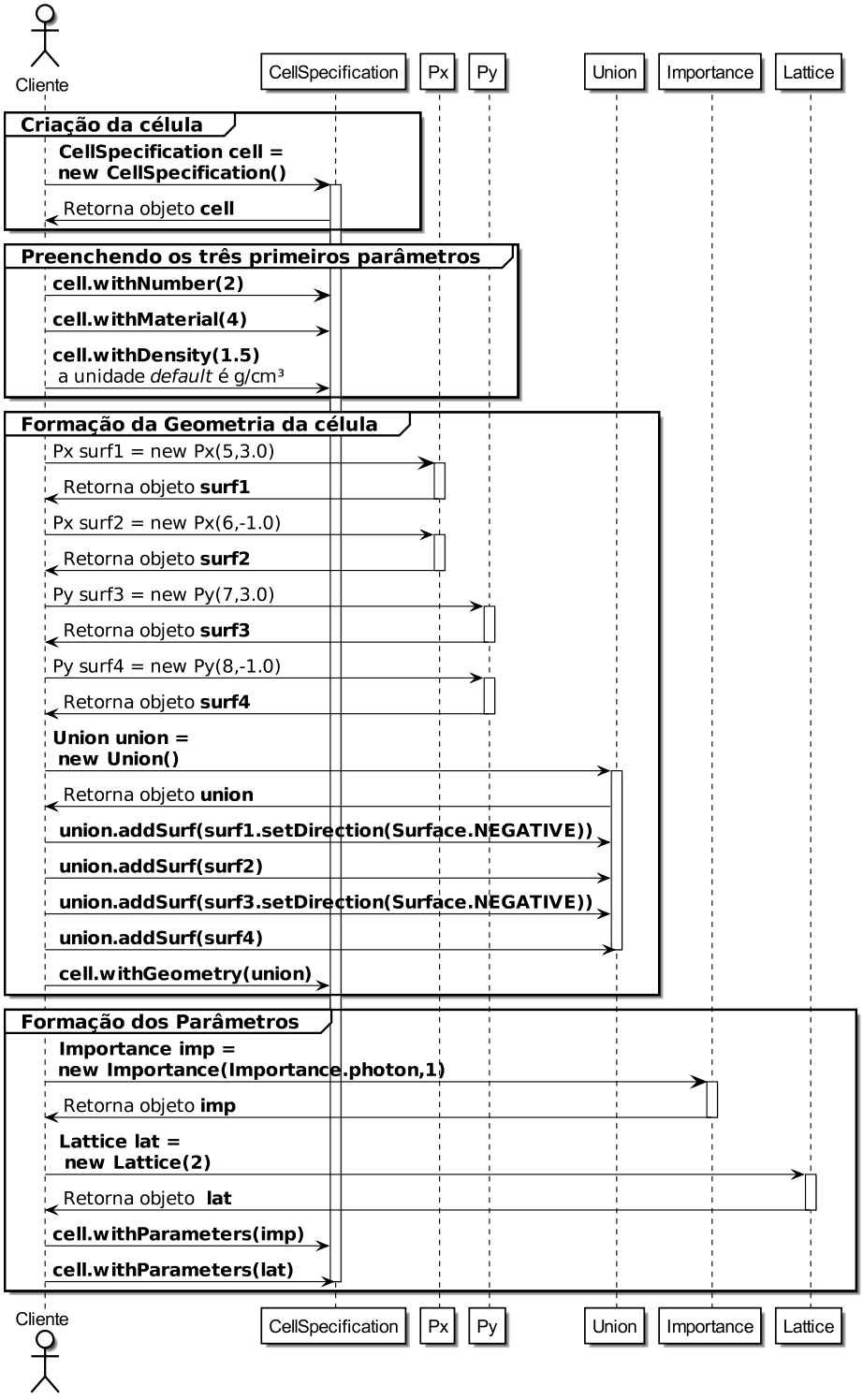

Figura C.28: Diagrama de sequência da criação de um célula conforme o segundo exemplo.

O terceiro exemplo é mostrado a seguir.

```
1  c celula 3
2  3  5  10.0  -9                imp:p=1
```

Onde os parâmetros de entrada são:

- Número de identificação igual à três;
- Número do material igual a cinco;
- Densidade igual a $10.0X10^{24} atoms/cm^3$;
- Geometria definida como a parte interior da superfície 9: "-9";
- Parâmetros de Célula: Importância, *IMP*, para fótons igual um;

Na Figura C.29 é mostrado o diagrama de sequência que mostra a criação dessa célula pela *helperMCNP*.

As diferenças nesse exemplo são a mudança de unidades da densidade pelo método *withRHOunit* e a geometria sendo composta por somente uma superfície. Conforme o primeiro exemplo, para extrair a representação dessa célula no *MCNP* é utilizado o método *draw*.

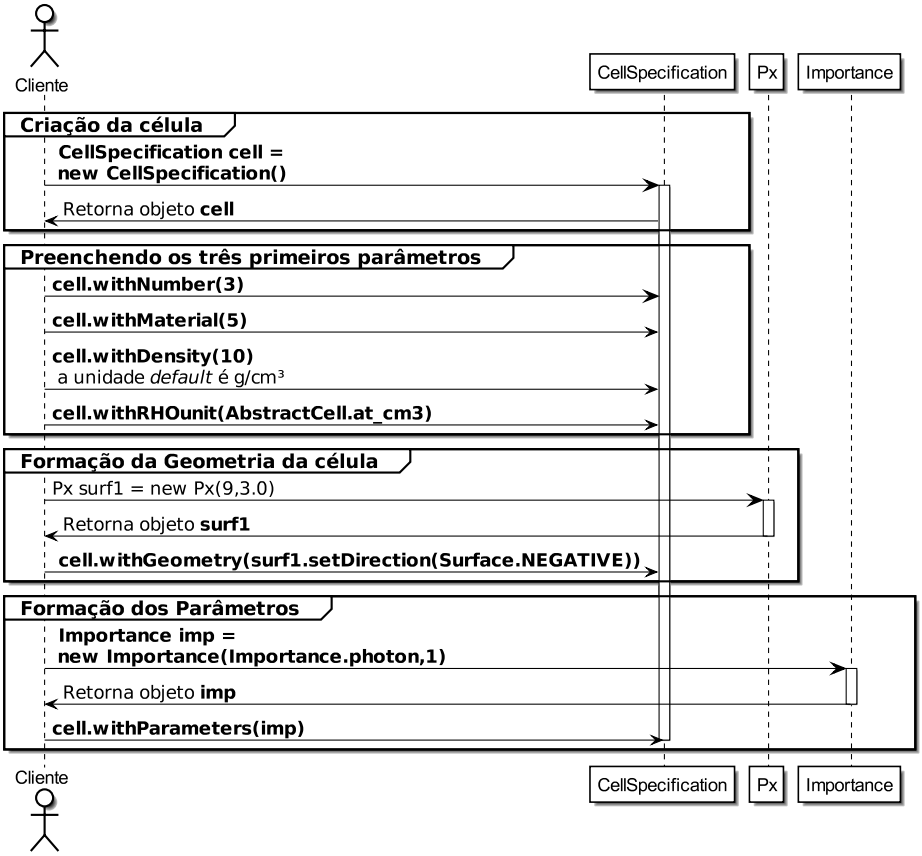

Figura C.29: Diagrama de sequência da criação de um célula conforme o terceiro exemplo.

O quarto exemplo é mostrado a seguir.

```
c celula 4
4  6   -2.0    #(1 2 -3)        fill=1 imp:e=2
```

Onde os parâmetros de entrada são:

- Número de identificação igual à quatro;
- Número do material igual a seis;
- Densidade igual a $2.0 g/cm^3$;
- Geometria igual a "#(1 2 -3)". Considera todas as regiões dentro do Universo 1 exceto às definidas pela intersecção das superfícies "1 2 -3";
- Parâmetros de Célula: Preenchido pelo *Universe* um mediante cartão *Fill* e Importância, *IMP*, para fótons igual um;

Na Figura C.30 há a o diagrama de sequência que mostra a criação dessa célula pela *helperMCNP*.

As diferenças nesse exemplo são o uso do complemento, para excluir um conjunto de superfícies e a utilização do parâmetro *Fill*. Conforme o primeiro exemplo, para extrair a representação dessa célula no *MCNP* é utilizado o método *draw*.

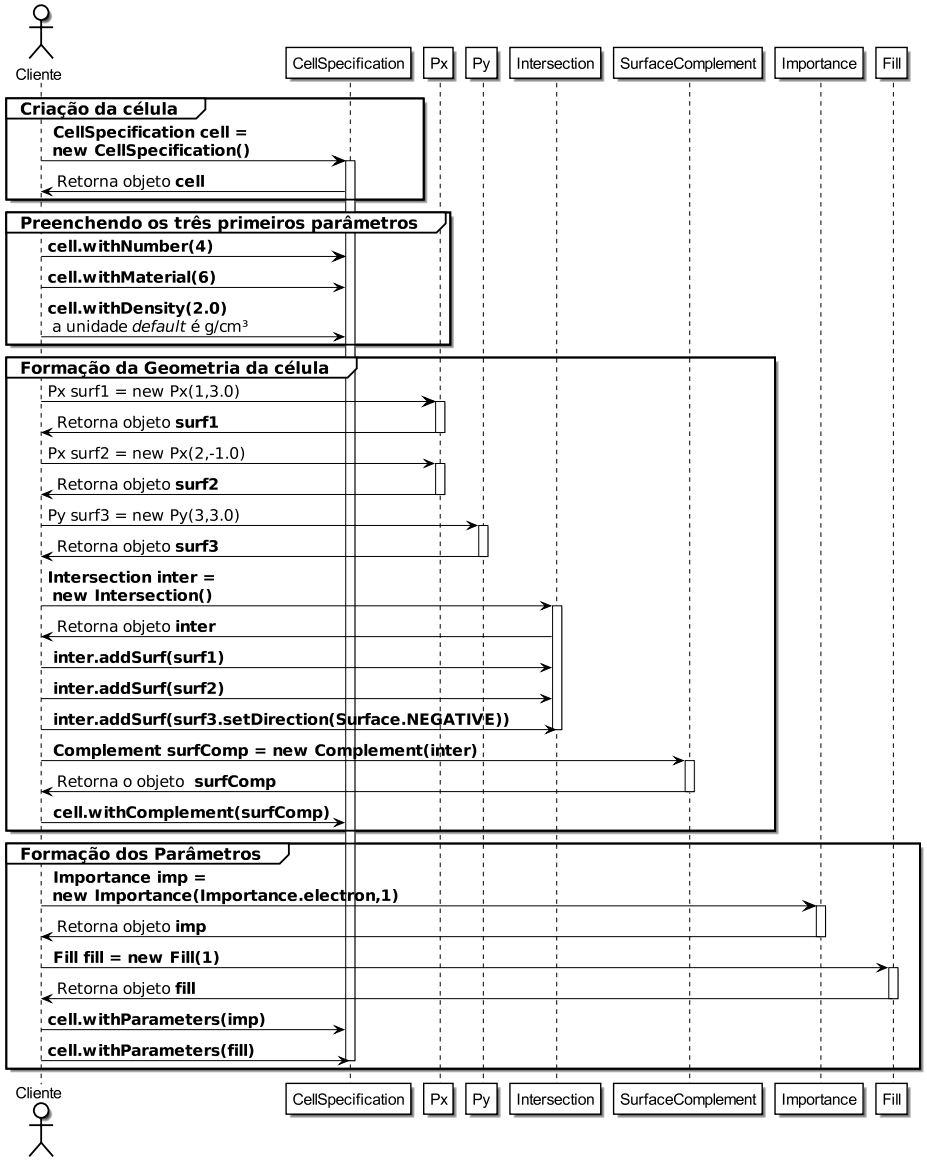

Figura C.30: Diagrama de sequência da criação de um célula conforme o quarto exemplo.

O quinto e último exemplo é mostrado a seguir.

```
1  c celula 5
2  5  7  -5.0  #2  fill 0:1 0:1 0:1
3                  1 4r 2 1r 1
```

Onde os parâmetros de entrada são:

- Número de identificação igual à cinco;
- Número do material igual a sete;
- Densidade igual a $5.0 g/cm^3$;

- Geometria igual a "#2". Exclui a célula dois dentre as células especificadas pelo cartão *Fill.*;

- Parâmetros de Célula: Preenchido por dois *Universe*, um e dois, mediante cartão *Fill* e Importância, *IMP*, para fótons igual um;

Na Figura C.31 é apresentado o diagrama de sequência que mostra a criação dessa célula pela *helperMCNP*.

As diferenças nesse exemplo são o uso do complemento para excluir uma célula e a utilização do parâmetro *Fill*, porém diferentemente do quarto exemplo, aqui ele tem a finalidade de preencher as células que compõem essa célula com diferentes *Universe*. É utilizado o método *draw* para extrair a representação dessa célula no *MCNP*.

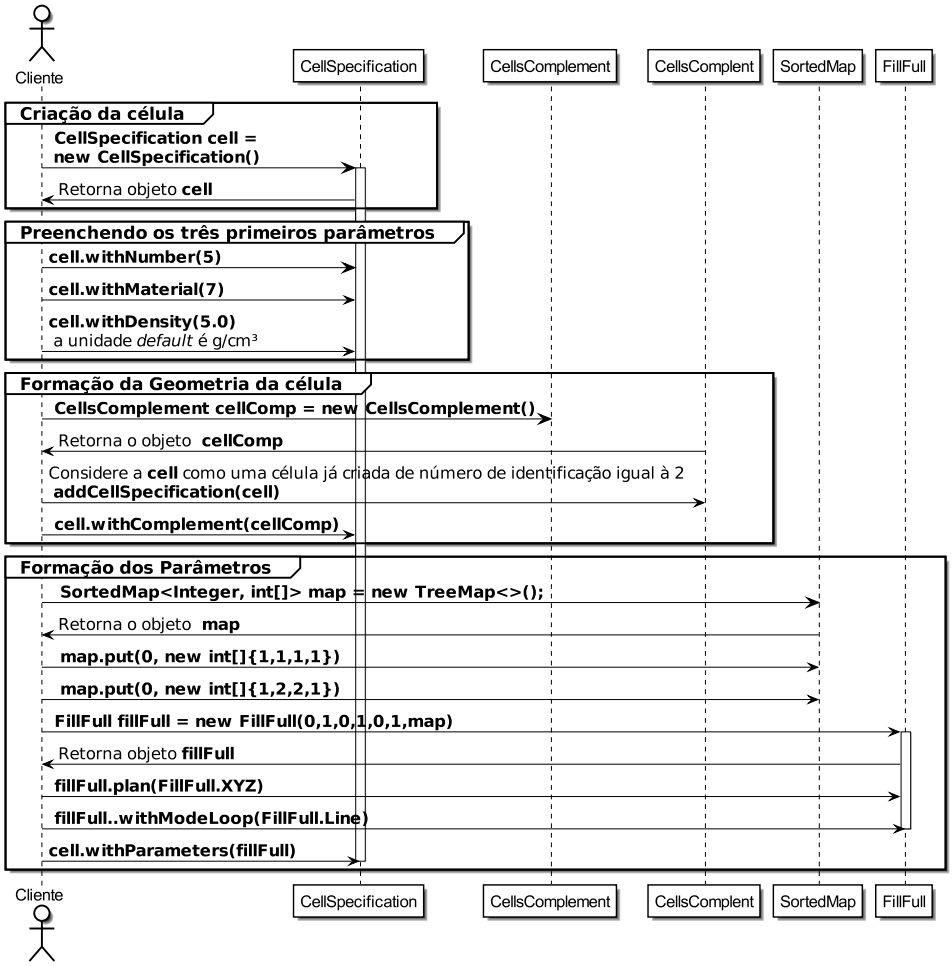

Figura C.31: Diagrama de sequência da criação de um célula conforme o quinto exemplo.

Declarar as células através do cartão "LIKE BUT": Além da maneira discutida anteriormente, há outra forma para se declarar uma célula no MCNP a qual se utiliza de

uma célula pré existente. Em termos gerais, ela permite que uma célula seja criada com base em outra, porém com alguns parâmetros diferentes.

Sua sintaxe no *MCNP* é "*j* ***LIKE*** *n* ***BUT*** *list*". Onde *j* é o número da célula, *n* é o número da célula que está sendo copiada e *list* é a lista de atributos que diferenciam essa célula da copiada. Em *list* há dois atributos importantes, um é o *MAT* que redefine o material dessa célula e o outro é o *RHO* que redefine a densidade da célula. Também é permitido declarar uma nova lista de parâmetros de célula para essa célula, mediante as variáveis *params* da Tabela C.7.

A seguir tem-se um exemplo de sua utilização em um código do *MCNP*.

```
2 4 -5.0     1  -2   3   -4   5   -6   imp:e=1
3 like 2 but mat=3 rho=-3.0 imp:p=2
```

Nesse exemplo a célula 3 é semelhante a célula 2 exceto por alguns parâmetros. Enquanto a célula 2 apresenta material e densidade igual a 4 e $5{,}0 g/cm^3$, respectivamente, a célula 3 modifica esses valores para 3 e $3{,}0 g/cm^3$. Outro diferencial da célula 3 é a adição de uma importância para fótons. É importante observar que a importância para elétrons continua a mesma para ambas as células.

Na Figura C.32 são mostradas as classes, pertencentes à biblioteca *helperMCNP*, responsáveis pela criação do cartão "LIKE BUT".

A classe *CellsLikeBut* recebe no construtor uma subclasse de *AbstractCell* que é a célula que se deseja copiar. Os demais parâmetros são semelhantes ao da classe *CellSpecification*. Para extrair sua representação é necessário executar o método *draw*.

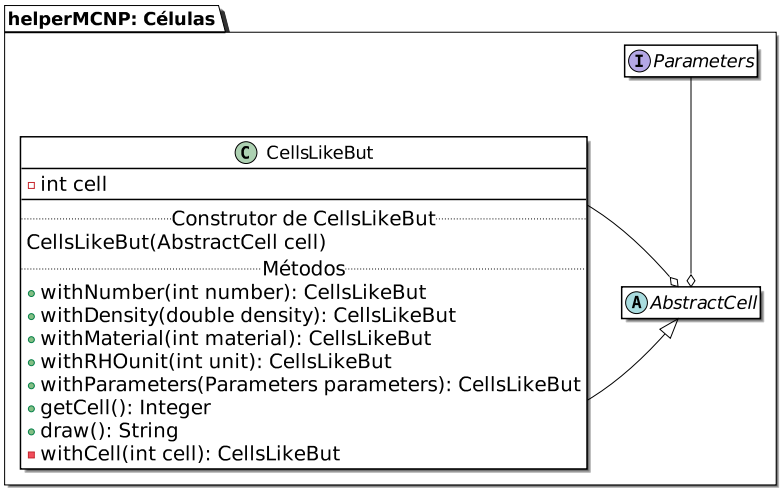

Figura C.32: Diagrama de classes referente à *CellsLikeBut*.

Para exemplificar o uso da classe *CellsLikeBut*, na Figura C.33, há um diagrama de sequência com o use da Classe *CellsLikeBut*.

Considerando, a célula *cellCopied* já forma, a *cell* referente à *CellsLikeBut* copiou-a e modificou alguns parâmetros. O primeiro foi trocar o material de dois para quatro depois trocou a densidade de $4.0 g/cm^3$ para $3.5 g/cm^3$. A seguir, foi adicionado o *Universe* à *cell*. A extração de sua representação foi feita pelo método *draw*.

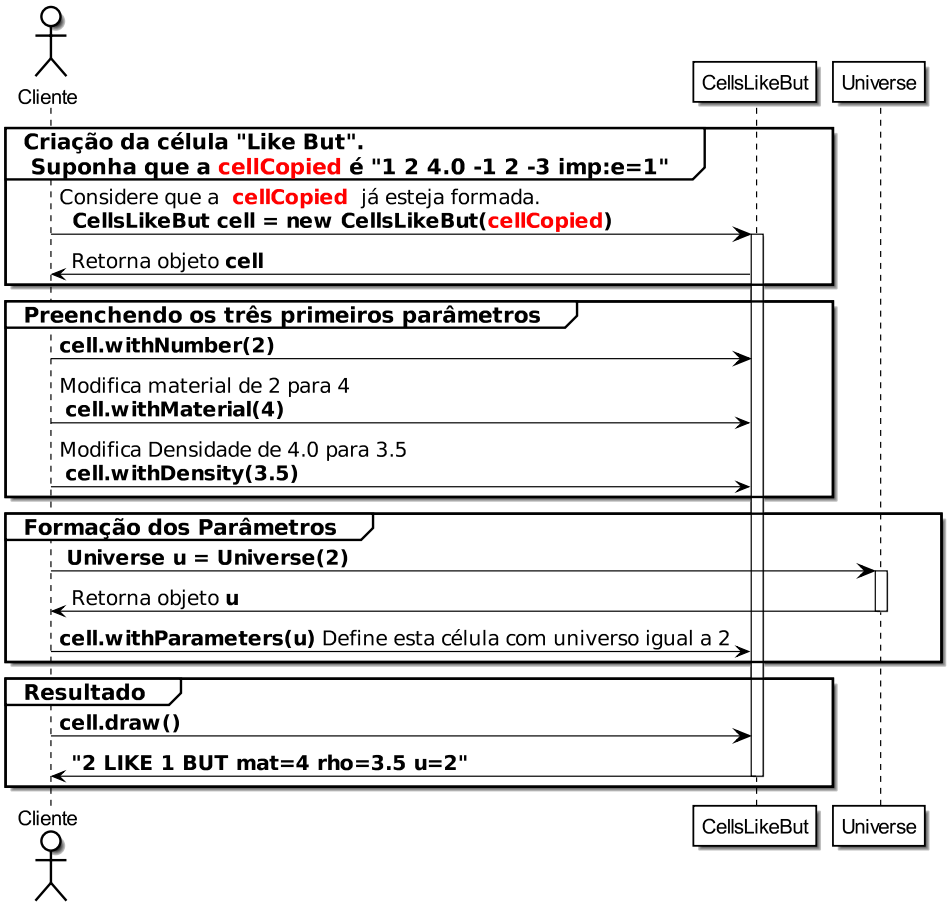

Figura C.33: Diagrama de sequência referente ao uso da classe *CellsLikeBut*.

C.1.3 Fonte de Radiação.

Há quatro formas de se definir uma fonte no *MCNP*, uma pelo cartão *SDEF* (Fonte Geral), outra pelo cartão *SSR* (Fonte de Superfície), outra pelo cartão *KCODE* (Fonte de Criticalidade) e por fim uma fonte fornecida pelo usuário.

Para construir as funções de distribuições dessas fontes, isto é, energia em função da sua probabilidade de ocorrência, o *MCNP* fornece os cartões *SI*(Informação da Fonte), *SP* (Probabilidade da Fonte), *SB* (Viés para a Probabilidade da Fonte) e *DS* (Distribuição de

Fonte Dependente).

Na biblioteca *helperMCNP* está somente implementado os cartões *SDEF*, *SSR*, *SI* e *SP* os quais são mais utilizados na simulação de sistemas de planejamento dosimétrico. Vale mencionar que a estrutura da biblioteca permite a inclusão dos demais cartões.

Antes de mencionar sobre os cartões *SDEF* e *SSR*, serão descritos os cartões *SI* e *SP*.

Cartões SI e SP: Os cartões *SI* e *SP* são frequentemente utilizados para representar, por exemplo, fontes polienergéticas, onde a fonte emite diversas energia com frequências diferentes. Seus usos não estão restritos somente à representação de fontes radioativas, de forma que podem ser utilizados em qualquer situação que se tenha a necessidade de especificar que uma determinada informação, *SI*, tenha uma probabilidade de ocorrência, *SP*.

Para entender como esses cartões foram implementados na biblioteca *helperMCNP*, na Figura C.34 é apresentado um diagrama de classes que envolve as classes *SI* e *SP*.

A interface *AbstractSourceS_IPB*, Figura C.34(a), representa uma abstração para as classes que irão criar os cartões *SI* e *SP*. Como será mencionado posteriormente esses cartões possuem mais de uma forma de serem representados, uma delas pode ser através de uma sequência de valores numéricos e em outros casos por uma sequência de caracteres padronizados. Por essa diversidade, foi escolhido abstrair o conceito dos valores contidos em *SI* e *SP*, mediante as interfaces, Figura C.34(b), *SI_Values*, para valores de *SI*, e *SP_Values*, para valores de *SP*. A classe *SI_SP* foi construída para facilitar a construção conjunta de *SI* e *SP*, pois os valores pertencentes as duas classes estão correlacionados, como será visto posteriormente.

O cartão *SI* possui a seguinte sintaxe no *MCNP*: "*SIn opção* i_1 ... i_k". A Tabela C.8 contêm as descrições dos parâmetros de *SI*. As informações dessa tabela foram retiradas do manual do *MCNP* (85).

Tabela C.8 - Descrição dos parâmetros do cartão *SI*.

Parâmetros	Descrições
n	O número da distribuição de *SI* que será especificada, por exemplo, no cartão *SDEF*. Restrição: $1 \leq j \leq 99.999$.

Continuação da Tabela C.8

Parâmetros	Descrições
opção	Determina como os valores de *SI* são interpretados. Há quatro *opções* possíveis H, L, A e S. A *opção* "H" indica que os valores representam um histograma. A *opção* "L" indica que os valores são discretos. A *opção* "A" indica pontos onde a PDF está definida. A *opção* "S" indica números que identificam outras distribuições. outras distribuições.
i_1 ... i_k	Valores de *SI*.

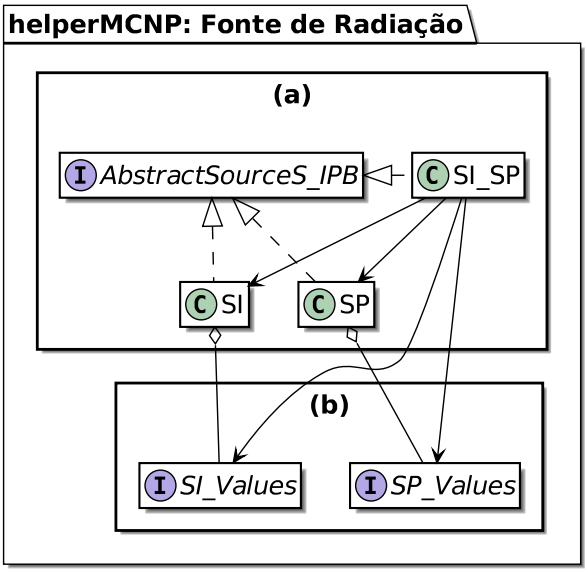

Figura C.34: Diagrama de classes de *SI*, *SP* e *SI_SP*.

Na Figura C.35 há um exemplo de dois tipos de representações possíveis do cartão *SI*. Na Figura C.35(a), é apresentada a maneira mais comum, em que o cartão *SI* contêm uma sequência de números referentes aos valores de *SI*. Figura C.35(b), ao invés dos números há uma sequência padronizada de grupos de caracteres, em que cada grupo representa a localização de um *Lattice* dentro de uma célula. Para definir o local do *lattice*, primeiramente tem-se que determinar qual a célula que o envolve, depois a célula que contêm o *lattice* juntamente com sua posição em relação aos demais e por fim a célula dessa posição. Além dessas duas, há também outras formas de especificar os valores de *SI*.

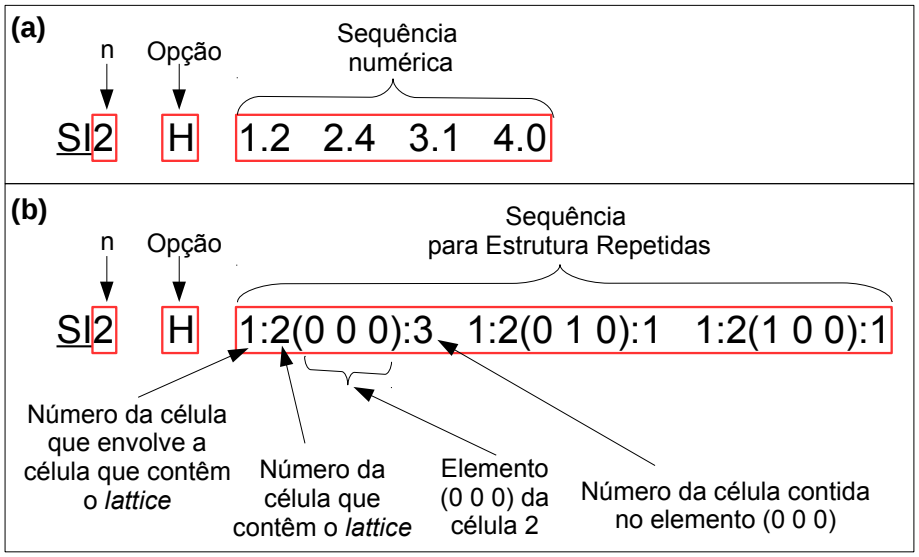

Figura C.35: Formas possíveis de representar o cartão *SI*.

Dessa forma, na Figura C.36, é apresentado o diagrama de classes que possibilita a criação do cartão *SI*.

A classe *SI* define entre os seus construtores o número de identificação da distribuição, o parâmetro da "opção" o qual possui seus valores como variáveis estáticas e por fim os valores de *SI* mediante subclasses de *SI_ Values*. A interface *SI_ Values* representa as diversas possibilidades de estruturar os valores de *SI*. As duas possibilidades implementadas foram *SI_ Number*, que representa os valores numéricos e *SI_ RepeatedStructured* que representa valores definidos pelas Estruturas Repetidas. É importante observar que *SI_ Values* requer das subclasses o método *getListString* que retorna a lista de valores porém em forma de *String* e o *getComparator* que retorna o modo como os dados foram ordenados em *SI*. A informação da ordenação dos valores de *SI* também serão necessárias para auxiliar a ordenação do cartão *SP* especificado posteriormente.

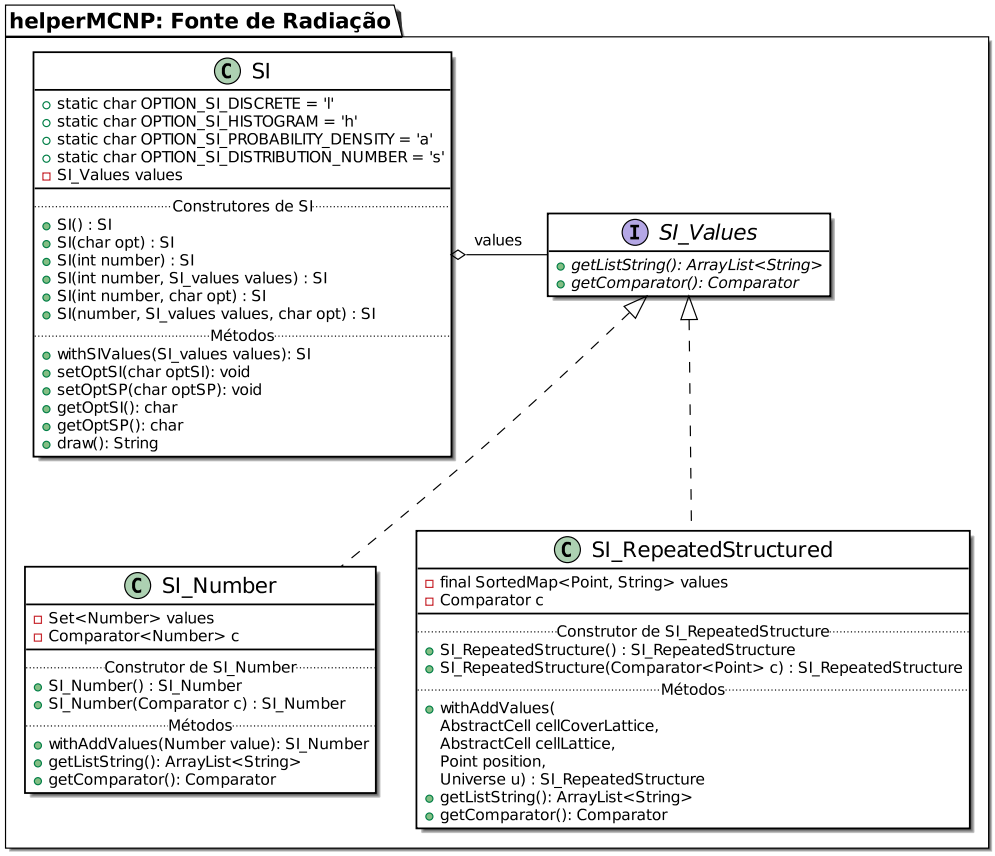

Figura C.36: Diagrama de classes de *SI* e sua relação com *SI_ Values* o qual representa os valores de *SI*.

A seguir, na Figura C.37, há um diagrama de sequência que exemplifica como é criado o cartão *SI* com valores numéricos, mediante a biblioteca *helperMCNP*.

Antes de criar uma instância da Classe *SI* é criada uma de *SI_ Number* que representa os valores numéricos de *SI*. No construtor de *SI_ Number* é possível passar como parâmetro um objeto de *Comparator* a fim de ordenar os valores de *SI* que serão adicionados. Os valores são adicionados pelo método *withAddValues* que recebe um objeto da classe *Number* que tanto pode ser um valor de inteiro ou de ponto flutuante. Após a criação de "***siValue***", objeto de *SI_ Number*, é criada o objeto de *SI* com a especificação do número da distribuição e da "opção" de valores discretos mediante o seu construtor. O objeto "***siValue***" poderia ser passado também nesse construtor, porém nesse exemplo ele é adicionado no objeto "***si***" através do método *withSIValues*. Para extrair a representação do objeto *SI* criado é utilizado o método *draw*.

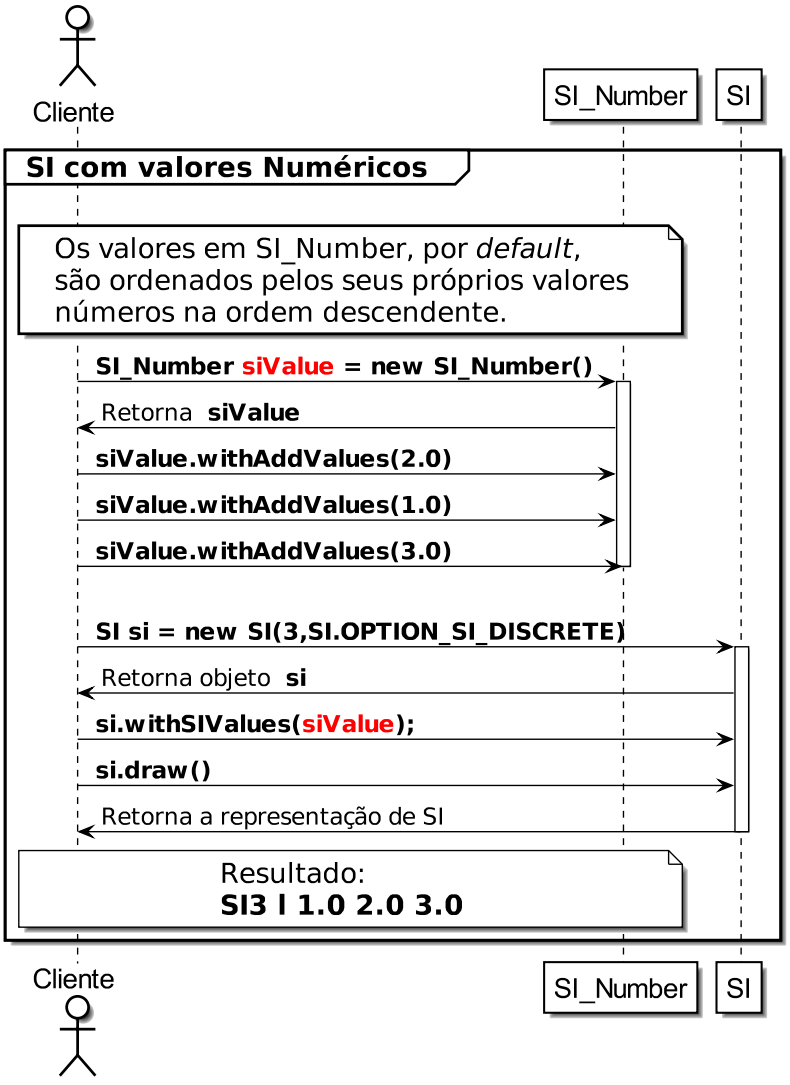

Figura C.37: Diagrama de sequência que exemplifica a criação do cartão *SI* com valores numéricos pela biblioteca *helperMCNP*.

A seguir, na Figura C.38, há um diagrama de sequência que exemplifica como é criado

o cartão *SI* com valores correspondentes às Estrutura Repetidas.

Antes de criar uma instância da Classe *SI* é criada uma de *SI_ RepeatedStructure* que representa os valores em relação às Estruturas Repetidas para *SI*. No construtor de *SI_ RepeatedStructure* é possível especificar um objeto de *Comparator* a fim de ordenar os valores de *SI* que serão adicionados.

Os valores são adicionados pelo método *withAddValues* que recebe quatro objetos. O primeiro é uma célula a qual cobre a célula do *lattice*, o segundo é a célula do *lattice*, o terceiro é a posição do *lattice* e por último é o objeto do universo desse *Lattice*.

A ordenação é feita através da localização do *lattice* expressa pela classe *Point*. Após a criação de "***siValue***", objeto de *SI_ RepeatedStructure*, é criada o objeto de *SI* com a especificação do número da distribuição e da "opção" de valores discretos mediante o seu construtor. O objeto "***siValue***" é adicionado no objeto "***si***" através do método *withSIValues*. Para extrair a representação do objeto *SI* criado é utilizado o método *draw*.

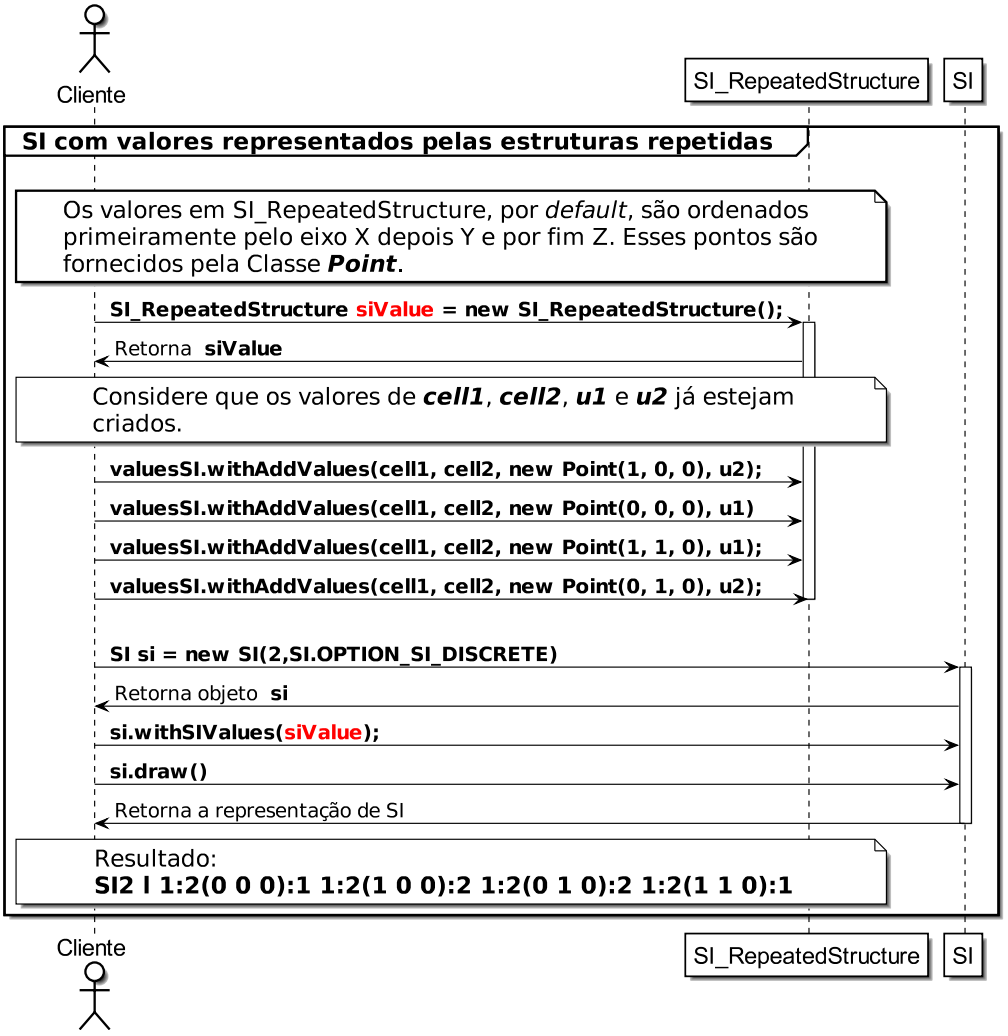

Figura C.38: Diagrama de sequência que exemplifica a criação do cartão *SI* com valores correspondentes às Estruturas Repetidas.

O cartão *SP* possui duas sintaxes, a primeira "*SPn opção* p_1 ... p_k" e a segunda "*SPn -f a b*", porém a biblioteca *helperMCNP* somente implementou a primeira sintaxe, pois a segunda não foi necessária para nenhuma simulação do sistema de planejamento. A Tabela C.9 contêm a descrição dos parâmetros da primeira sintaxe. As informações dessa tabela foram retiradas do manual do *MCNP* (85).

Tabela C.9 - Descrição dos parâmetros do cartão SP.

Parâmetros	Descrições
n	O número da distribuição que corresponderá ao do *SI*.
	Restrição: $1 \leq j \leq 99.999$.

Continuação da Tabela C.9

Parâmetros	Descrições
opção	Determina como os valores de *SP* são interpretados. Há quatro *opções* possíveis D, C, V e W. A *opção* "D" indica que os valores de *SP* são intervalos de probabilidade para as opções H e L de *SI*. No *MCNP* esta opção é a padrão. A *opção* "C" indica que os valores são intervalos acumulativos de probabilidades para as opções H e L de *SI*. A *opção* "V" indica que os valores são distribuições de células e a probabilidade é proporcional ao volume da célula. A *opção* "W" indica que os valores são intensidades para uma mistura de partículas de fonte.
$p_1 \ \ldots \ p_k$	Valores de *SP*. Restrições: Caso for um Histograma, H, o primeiro valor deve ser zero.

Na Figura C.39 há um exemplo do cartão *SP* e sua correlação com *SI*.

Tanto na Figura C.39(a) como na Figura C.39(b), o número da distribuição *SI* é o mesmo que a do *SP* indicando que seus respectivos valores estarão correlacionados. Isto é, cada valor de *SI* possui um valor de *SP*, indicado na figura pelas setas. Está correlação é feita através das posições de cada um dos valores dentro da sequência. A diferença entre as situações representadas na Figura C.39(a) e Figura C.39(b), está relacionada com a forma como que os dados de *SI* são ordenados. Na Figura C.39(a) os dados de *SI* são ordenados através de seus valores, já na Figura C.39(b) os dados são ordenados mediante a posição do *Lattice*. E é com base na ordenação de *SI* que os valores de *SP* são ordenados.

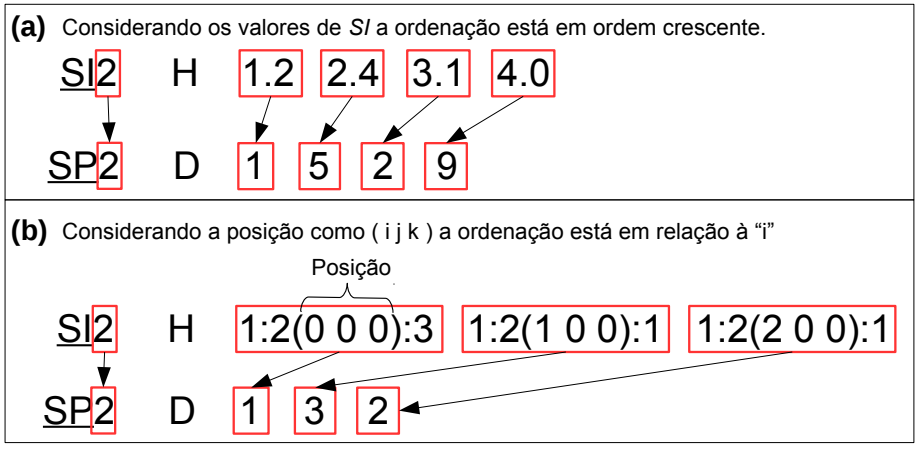

Figura C.39: Relação entre o cartão *SP* e *SI*.

Na Figura C.40 é apresentado o diagrama de classes referente ao *SP*.

A classe *SP* define entre os seus construtores o número de identificação da distribuição, o parâmetro da "opção" o qual possui seus valores como variáveis estáticas e por fim os valores de *SP* mediante subclasses de *SP_ Values*. A interface *SP_ Values* representa os diferentes tipos de valores de *SP*. A diferença entre *SP_ Number* e *SP_ Number_ RepeatedStructure* é o modo como os dados são ordenados. Em *SP_ Number* os valores são ordenados segundo valores numéricos fornecidos pelo *SI* e em *SP_ Number_ RepeatedStructured* os valores são ordenados segundo a localização do *lattice* fornecida por *SI* mediante objetos da classe *Point*.

Assim como em *SI_ Values*, *SP_ Values* possui o método *getListString* a fim de fornecer a sequência de valores de *SP* na forma de *String*.

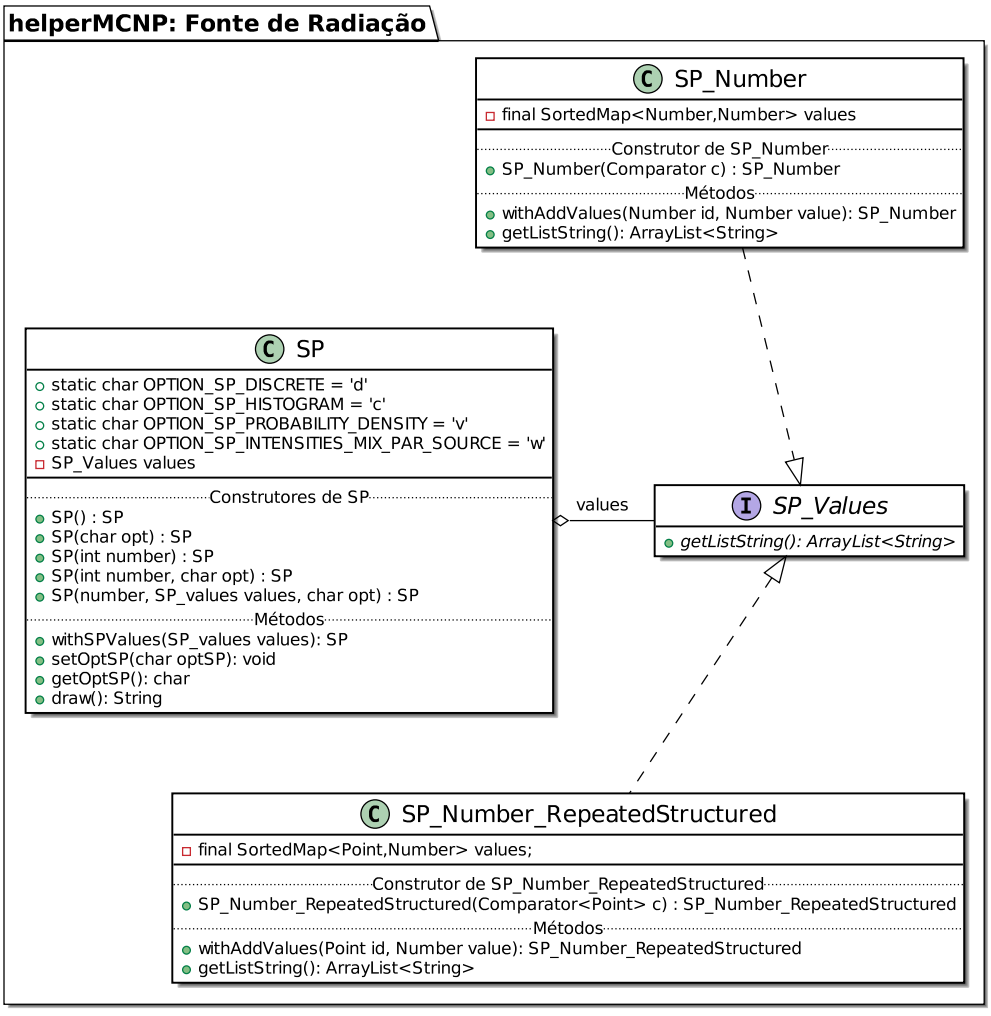

Figura C.40: Diagrama de classes de *SP* e sua relação com *SP_ Values* o qual representa os valores de *SP*.

Na Figura C.41 há um diagrama de sequência que exemplifica o uso da classe *SP* a fim de criar um cartão *SP* com seus valores correlacionados com os valores numéricos de *SI*.

O objeto de *Comparator*, parâmetro do construtor de *SP_Number* especifica a ordenação dos valores de *SP*. Os valores são adicionados pelo método *withAddValues* que recebe dois objetos da classe *Number*. O primeiro é o valor numérico de *SI* e o segundo é o valor numérico de *SP*.

A ordenação é feita com o uso dos valores numéricos de *SI*. Após a criação de "***spValue***", objeto de *SI_Number*, é criada o objeto de *SP* com a especificação do número da distribuição e da "opção" de valores discretos mediante o seu construtor.

O objeto "***spValue***" é adicionado no objeto "***sp***" através do método *withSIValues*. Para extrair a representação do objeto *SP* criado é utilizado o método *draw*.

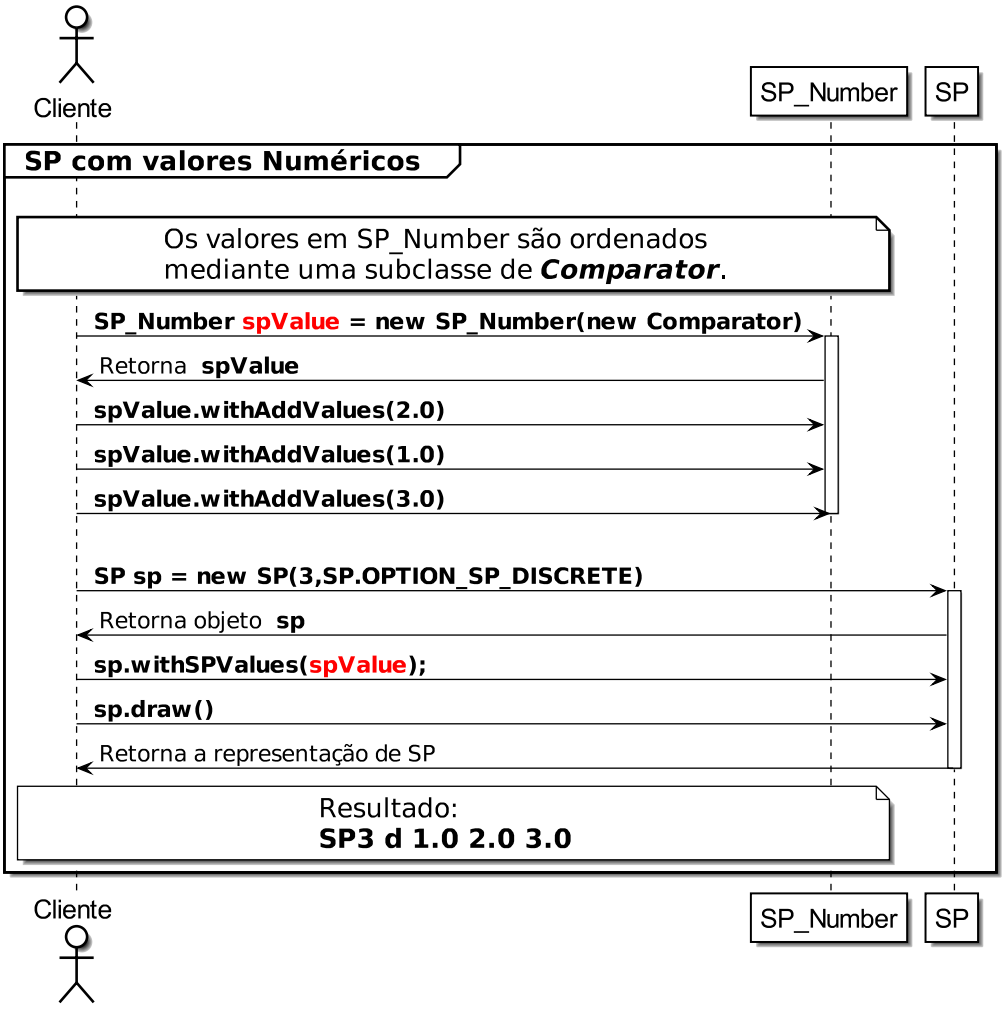

Figura C.41: Diagrama de sequência que exemplifica a criação do cartão *SP* com seus valores correlacionados com os valores numéricos de *SI*.

Outro exemplo com o uso da classe *SP* esta na Figura C.42, em que representa um

caso onde os valores de *SP* são referentes aos valores de *SI* formados com representação de Estrutura Repetidas.

Primeiramente é criada a classe *SP_Number_RepeatedStructure*. O *Comparator* passado como parâmetro para o construtor de *SP* ordena valores da classe *Point*.

Os valores, em *SP_Number_RepeatedStructure*, são adicionados pelo método *withAddValues* que recebe dois objetos um da clase *Point* e o outra da classe *Number*. O primeiro é a localização do *lattice* especificada nos valores de *SI* e o segundo é o valor numérico de *SP*.

Após a criação de "***spValue***", objeto de *SP_Number_RepeatedStructure*, é criado o objeto de *SP* com a especificação do número da distribuição e da "opção" de valores discretos mediante o seu construtor.

O objeto "***spValue***" é adicionado no objeto "***sp***" através do método *withSIValues*. Para extrair a representação do objeto *SP* criado é utilizado o método *draw*.

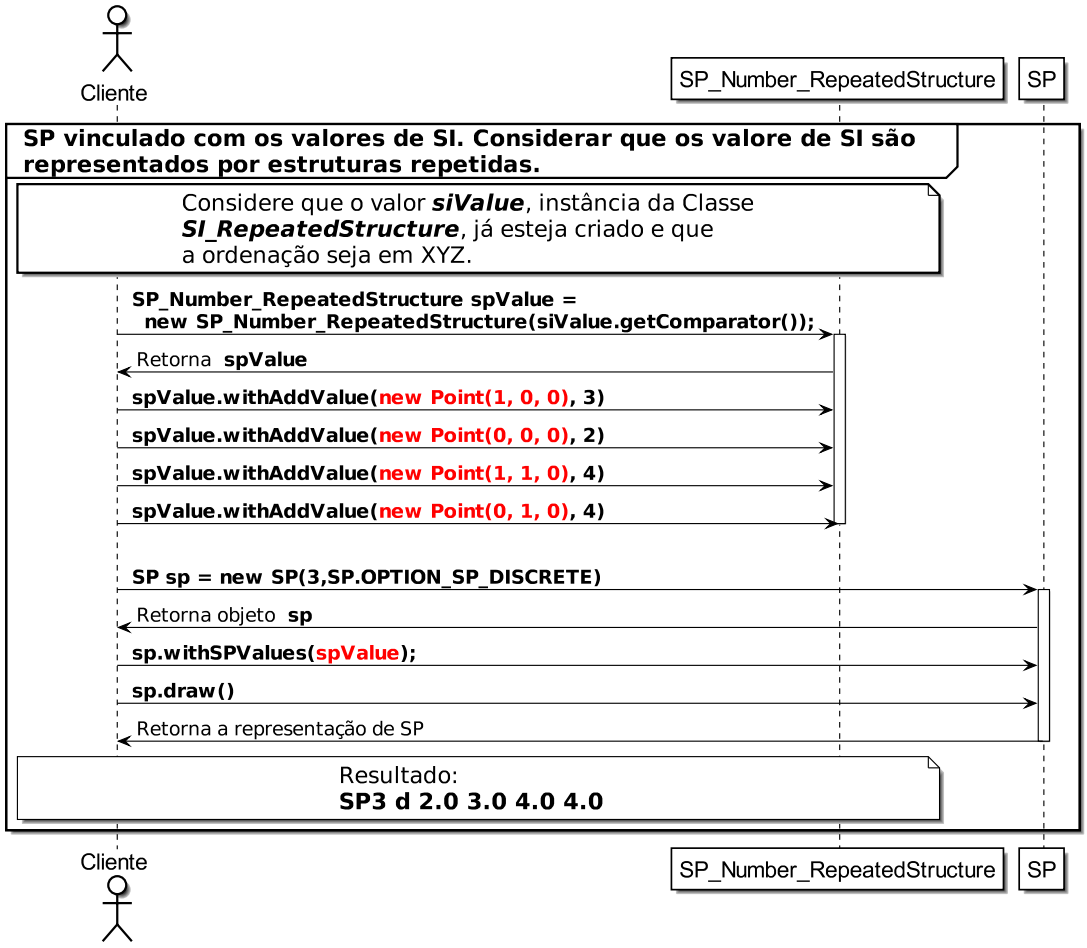

Figura C.42: Diagrama de sequência que exemplifica a criação do cartão *SP* com seus valores correlacionados aos valores de *SI* especificados com relação à Estruturas Repetidas.

Como praticamente todos os casos simulados especificam valores de *SI* vinculados aos valores de *SP* então foi criada a classe *SI_SP*. Na Figura C.43 há o diagrama de classes de *SI_SP*.

A classe *SI_SP* define entre os seus construtores o número de identificação da distribuição, os valores de *SI* mediante subclasses de *SI_Values* e os valores de *SP* mediante subclasses de *SP_Values*.

Há duas possibilidades de se criar um objeto da classe *SI_SP*, a primeira é especificar *SI_Number* e *SP_Number* outra é especificar *SI_RepeatedStructure* e *SP_Number_RepeatedStructure*. Por intermédio do método *draw* é possível adquirir a representação de *SI* e *SP*. A classe *SI_SP*, assim como *SI* e *SP*, é subclasse de *AbstractSourceS_IPB*.

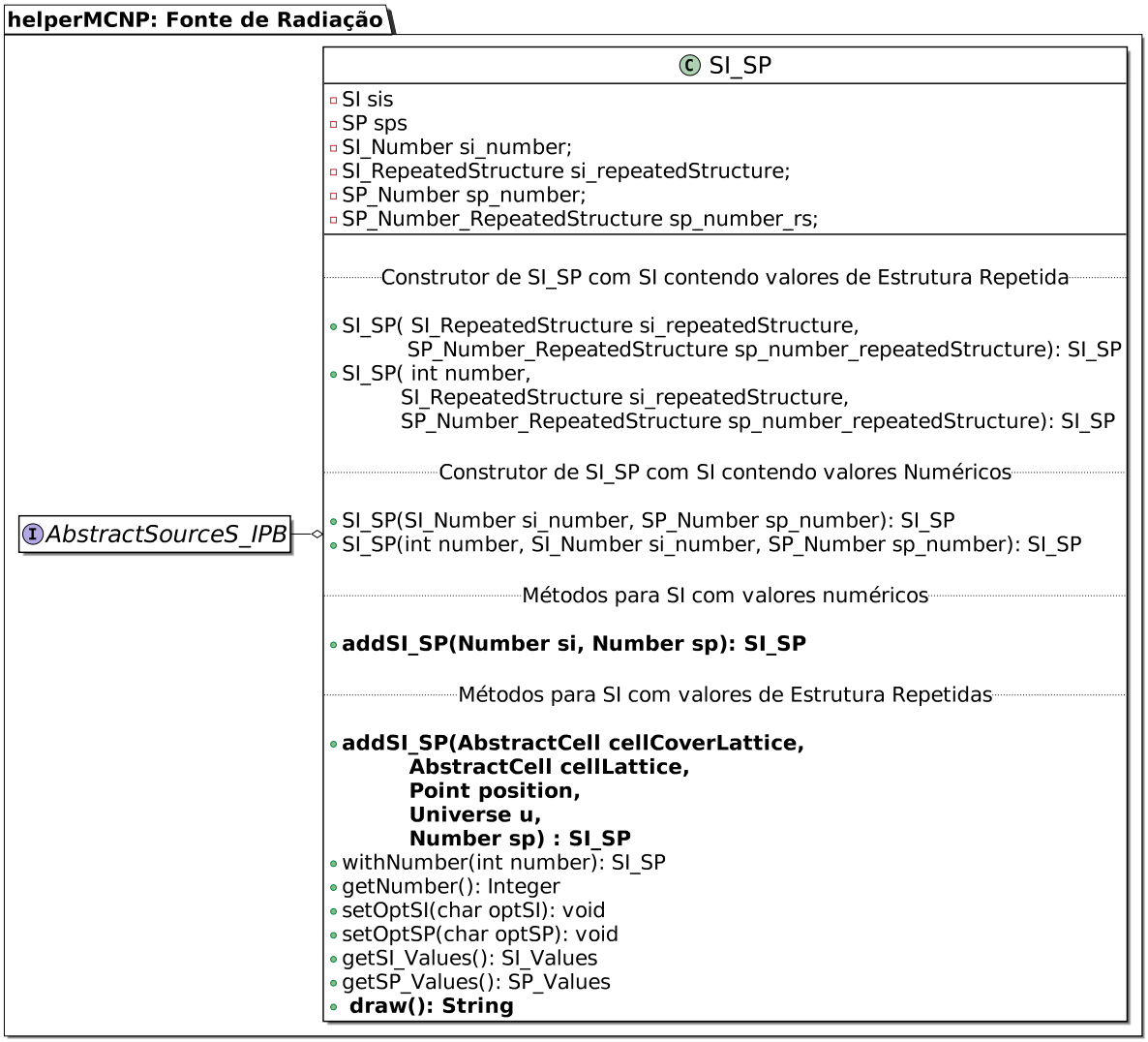

Figura C.43: Diagrama de classes de *SI_SP*.

Para exemplificar o uso da classe *SI_SP* a Figura C.44 apresenta um diagrama de

sequência da criação de *SI_SP*.

Primeiramente são criados os objetos de *SI_Number*,***siValue***, e *SP_Number*, ***spValue***, porém sem a necessidade de preenchê-los com seus respectivos valores.

É importante observar que o objeto ***siValue*** é criado antes do ***spValue*** e isto se deve pois o método *getComparator* de ***siValue*** será utilizado para construir o ***spValue***. Dessa forma, os valores de *SP* estarão ordenados conforme os do *SI*.

Após a construção desses dois objetos é criado o objeto ***si_sp*** que recebe em seu construtor o número de identificação da distribuição e os dois objetos criados anteriormente. Com o método *addSI_SP(Number id, Number sp)* são adicionados os valores de *SI* e *SP* como números. Para extrair a representação do objeto ***si_sp*** criado é utilizado o método *draw*.

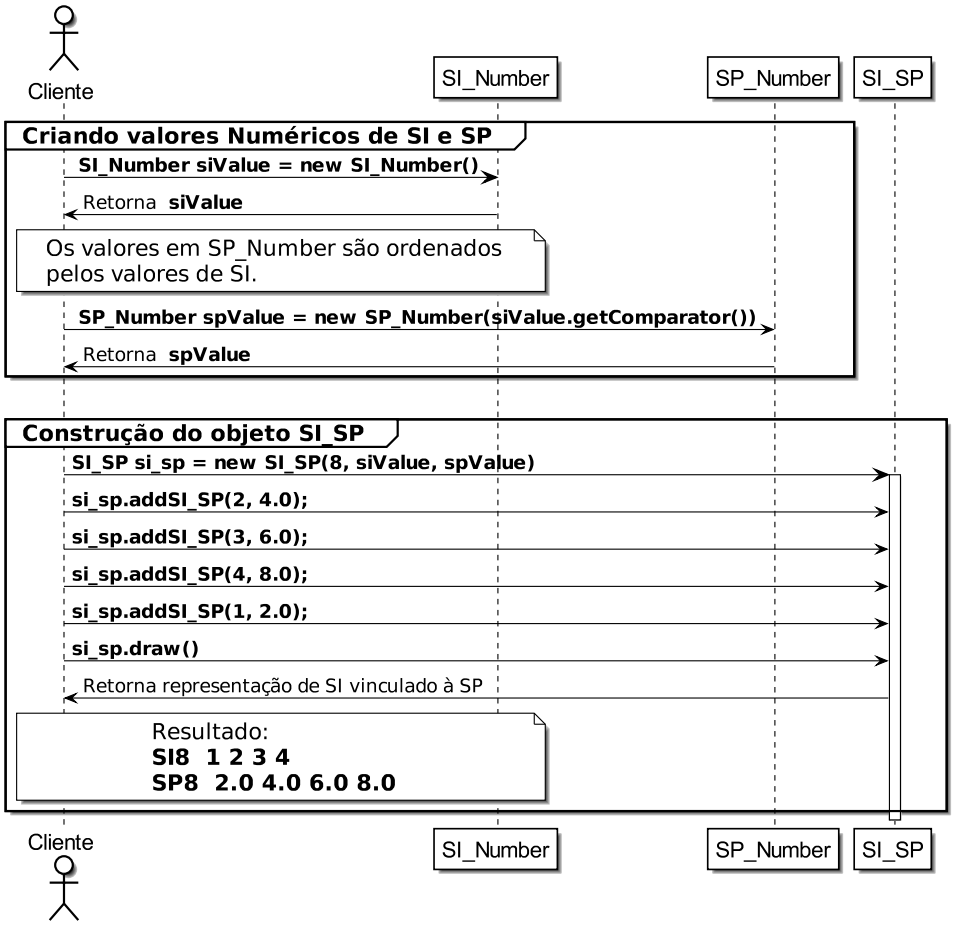

Figura C.44: Diagrama de sequência que exemplifica o uso da classe *SI_SP* onde tanto os valores de *SI* como *SP* são números.

Na Figura C.45 há outro exemplo de *SI_SP* porém nesse caso ele foi construído para valores de *SI* mediante representação de Estruturas Repetidas.

Primeiramente são criados os objetos de *SI_ RepeatedStructure*,***siValue***, e *SP_ Number_ RepeatedStructure*, ***spValue***.

Para criar o objeto ***spValue*** foi utilizado o método *getComparator* de ***siValue***, que nesse caso retorna um comparador de pontos definidos pela classe *Point*. Após a construção desses dois objetos é criado o objeto ***si_ sp*** que recebe em seu construtor o número de identificação da distribuição e os dois objetos criados anteriormente.

Com o método *addSI_ SP(AbstractCell cell1, Abstract cell2, Point position, Universe u, Number sp)* são adicionados os valores de *SI* e *SP*. Para extrair a representação do objeto ***si_ sp*** criado é utilizado o método *draw*.

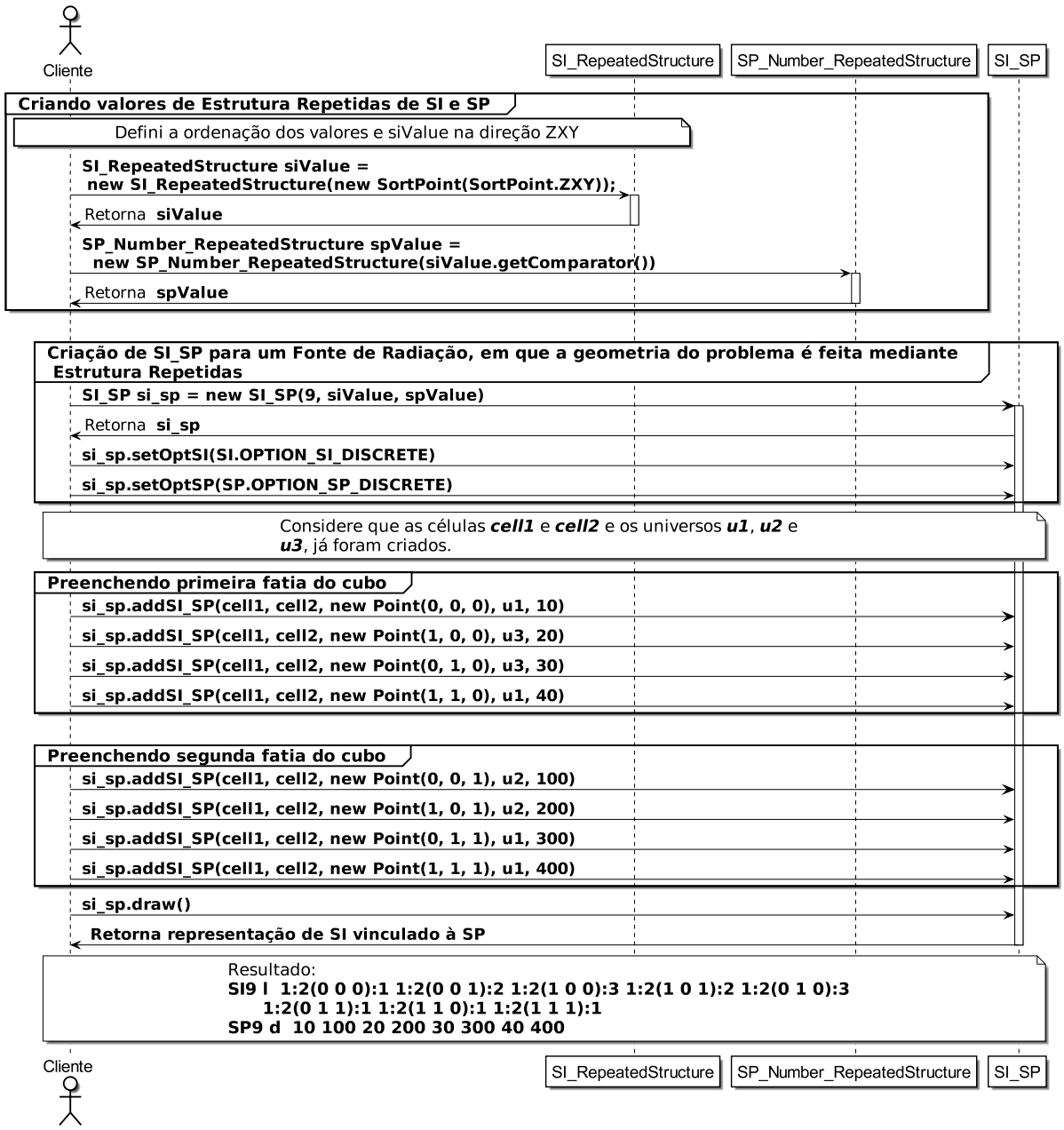

Figura C.45: Diagrama de sequência que exemplifica o uso da classe *SI_SP* onde os valores de *SI* são representados com base em Estruturas Repetidas.

Cartão SDEF: O cartão *SDEF* é utilizado para especificar uma fonte no *MCNP*. Ele possui a seguinte sintaxe "*SDEF parâmetro=valore(s) ...* ". A Tabela C.10 apresenta somente os parâmetros, de *SDEF*, que foram implementados em *helperMCNP*. As informações dessa tabela foram retiradas do manual do *MCNP* (85).

Tabela C.10 - Descrição dos parâmetros do cartão *SDEF* que estão contidos em *helperMCNP*.

Parâmetros	Descrições
CEL	O número da célula. Considera que a fonte esteja contida dentro de uma célula.

Continuação da Tabela C.10

Parâmetros	Descrições
ERG	Energia da Fonte em MeV.
DIR	Ângulo azimutal entre seu vetor referência e a direção de voo das partículas.
VEC	Vetor referência de *DIR*.
X	Posição da Fonte na coordenada X.
Y	Posição da Fonte na coordenada Y.
Z	Posição da Fonte na coordenada Z.

Na Figura C.46 há o diagrama de classes referente ao cartão *SDEF*. A classe responsável por criar o cartão *SDEF* possui este mesmo nome. Alguns parâmetros, por exemplo *ERG*, podem ter um único valor ou serem representados por uma distribuição mediante *SI* e *SP*.

Dessa forma esses parâmetros possuem dois métodos, um para especificar um único valor e o outro para especificar uma distribuição, sendo que neste último, a distribuição passada será uma subclasse de *AbstractSourceS_IPB*.

Para extrair a representação de *SDEF* é necessário requerer o método *draw*.

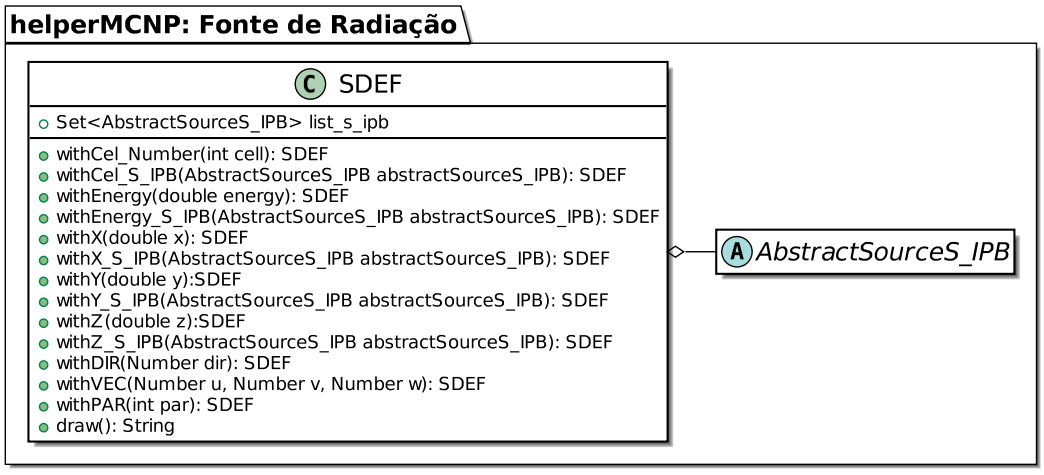

Figura C.46: Diagrama de classes referente ao cartão *SDEF*.

Para exemplificar o uso da classe *SDEF* foi elaborado na Figura C.47 o diagrama de sequência referente a criação de uma fonte no *MCNP*.

O exemplo, corresponde a criação de uma fonte de fótons, localizada numa célula de número de identificação igual à 2 e com o espectro energético (Energia,Frequência) igual à (1.0,20), (2.0,30) e (3.0,50).

Primeiramente, para se criar o espectro energético é utilizada a classe *SI* e *SP* com valores numéricos, discutidas anteriormente. Após a criação dos dois objetos dessas classes,

eles são adicionados no parâmetro *ERG* através do método *withEnergy_S_IPB* que recebe uma subclasse de *AbstractSourceS_IPB*.

Para especificar qual célula a fonte se encontra, é utilizado o método *withCel_Number* que recebe o número de identificação da célula. Por fim para especificar qual partícula é emitida da fonte é utilizado o método *withPAR* que recebe um número de partícula especificado como constante no *MCNP*.

Para facilitar a lembrança dos números das partículas foi criada a classe *Constants* a qual apresenta, em forma de variáveis estáticas, alguns desses valores de partículas. Como pode ser observado para requerer o número dos fótons foi utilizada a variável estática *Constants.PAR_PHOTON*. Para adquirir a representação dessa fonte foi usado o método *draw*. O resultado é apresentado no final do diagrama.

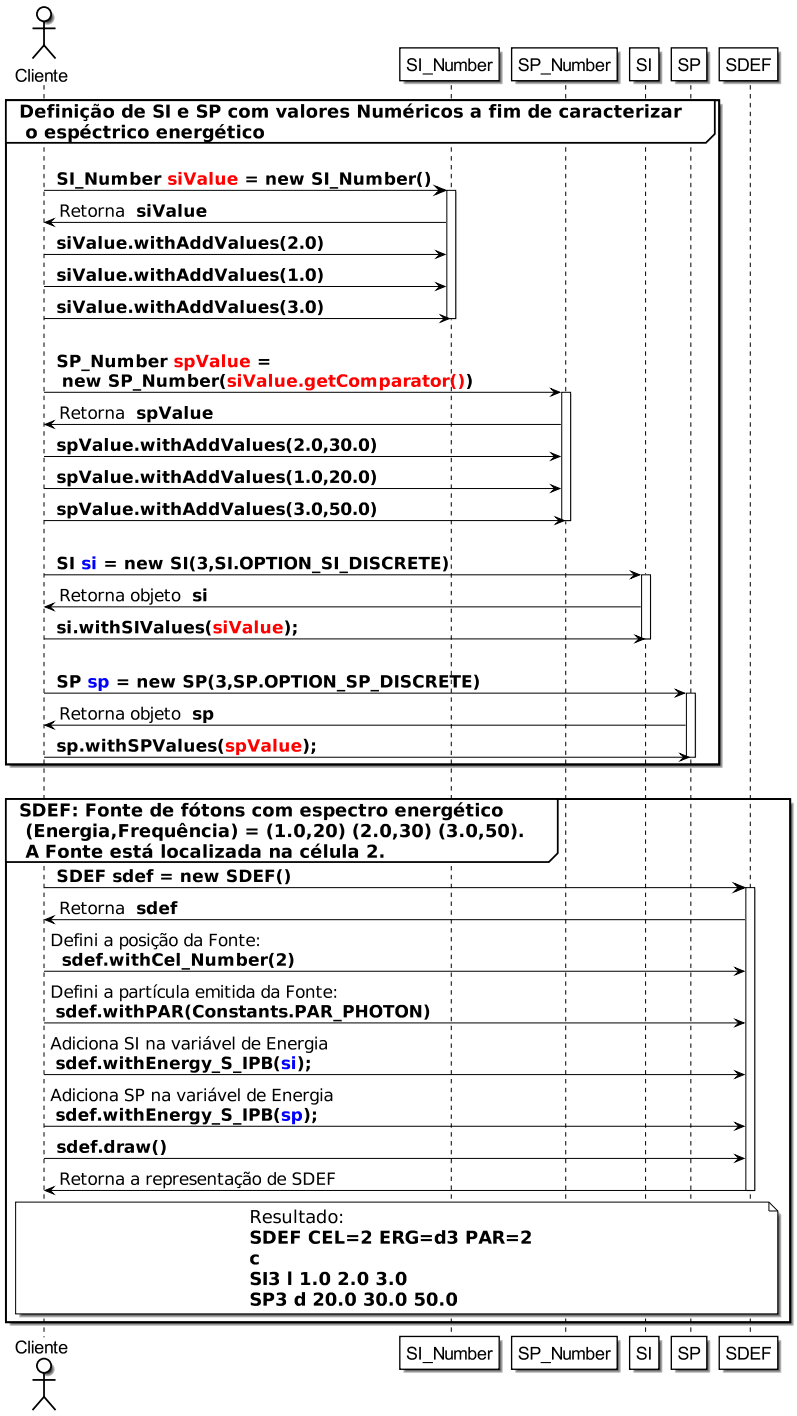

Figura C.47: Diagrama de sequência que corresponde a criação de uma fonte de fótons.

Cartão SSR: O *MCNP* disponibiliza o cartão *SSW*,*Surface Source Write*, para possibilitar o registro de todas as partículas que atravessam determinadas superfícies. O arquivo gerado é chamado de espaço de fase que pode ser usado posteriormente em substituição a fonte original. Esse processo gera um arquivo binário que poderá ser lido por um outro arquivo *INP* do *MCNP*. O cartão utilizado para ler a informação contida no arquivo gerado

pelo *SSW* é o *SSR*, *Surface Source Read*.

A sintaxe desse cartão no *MCNP* é : "*SSR parâmetros=valore(s) ...* ". Na Tabela C.11 há os parâmetros desse cartão que foram implementos na biblioteca *helperMCNP*. As informações dessa tabela foram retiradas do manual do *MCNP* (85).

Tabela C.11 - Descrição dos parâmetros do cartão *SSR* que estão contidos em *helperMCNP*.

Parâmetros	Descrições
OLD	As superfícies da fonte de radiação simulada pelo cartão *SSW*.
TR	Especifica uma transformação em relação à fonte de radiação. Isto é, modifica o posicionamento da fonte. Pode receber uma distribuição de *SI* e *SP* a fim de especificar diversar transformações da fonte.
WGT	Os pesos das partículas são multiplicados pela constante *WGT*.

Na Figura C.48 é apresentado o diagrama de classes que envolve a criação do cartão *SSR* pela biblioteca *helperMCNP*.

A classe responsável por essa criação é a *SSR*. No cartão *SSR* há alguns parâmetros, por exemplo *TR*, os quais podem ser representados por distribuições mediante os cartões *SI* e *SP*.

Como a classe *SSR* implementa o parâmetro *TR*, ela possui um método que auxilia em sua criação quando é preciso especificar uma distribuição em *TR*, *addS_IPB_TR*. Há também a presença da classe *TR*, no diagrama, que foi criada pelo fato desse parâmetro não ser específico somente à esse cartão.

Há também um vínculo entre *SSR* e *Surface* mediante o parâmetro *OLD* o qual recebe um lista de objeto das subclasses de *Surface*. Para gerar a representação de *SSR* para o *MCNP* é utilizado o método *draw*.

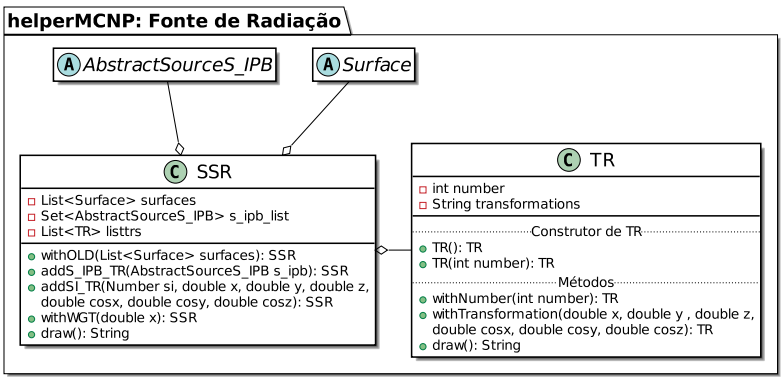

Figura C.48: Diagrama de classes referente à criação do cartão *SSR*.

Para exemplificar o uso da classe *SSR* foi elaborado o diagrama de classes apresentado na Figura C.49.

Considere primeiramente que a fonte de radiação já foi criada pelo cartão *SSW*. Considere também que as superfícies, *surf1*, *surf2* e *surf3* referente às fontes também já foram criadas.

A distribuição de *TR* é criada mediante os cartões *SI* e *SP* a fim de especificar que a distribuição de translação da fonte de radiação. Essa distribuição é vinculada ao parâmetro *TR* através do método *addS_IPB_TR*.

Após esse vínculo é especificada os parâmetros de *TR* à cada valor de *SI*. Com o método *withOLD* as superfícies da fonte são acrescentadas no cartão *SSR*. Por fim através do método *withWGT* é especificado o parâmentro *WGT*. No fim do diagrama é apresentado o resultado gerado pelo método *draw*.

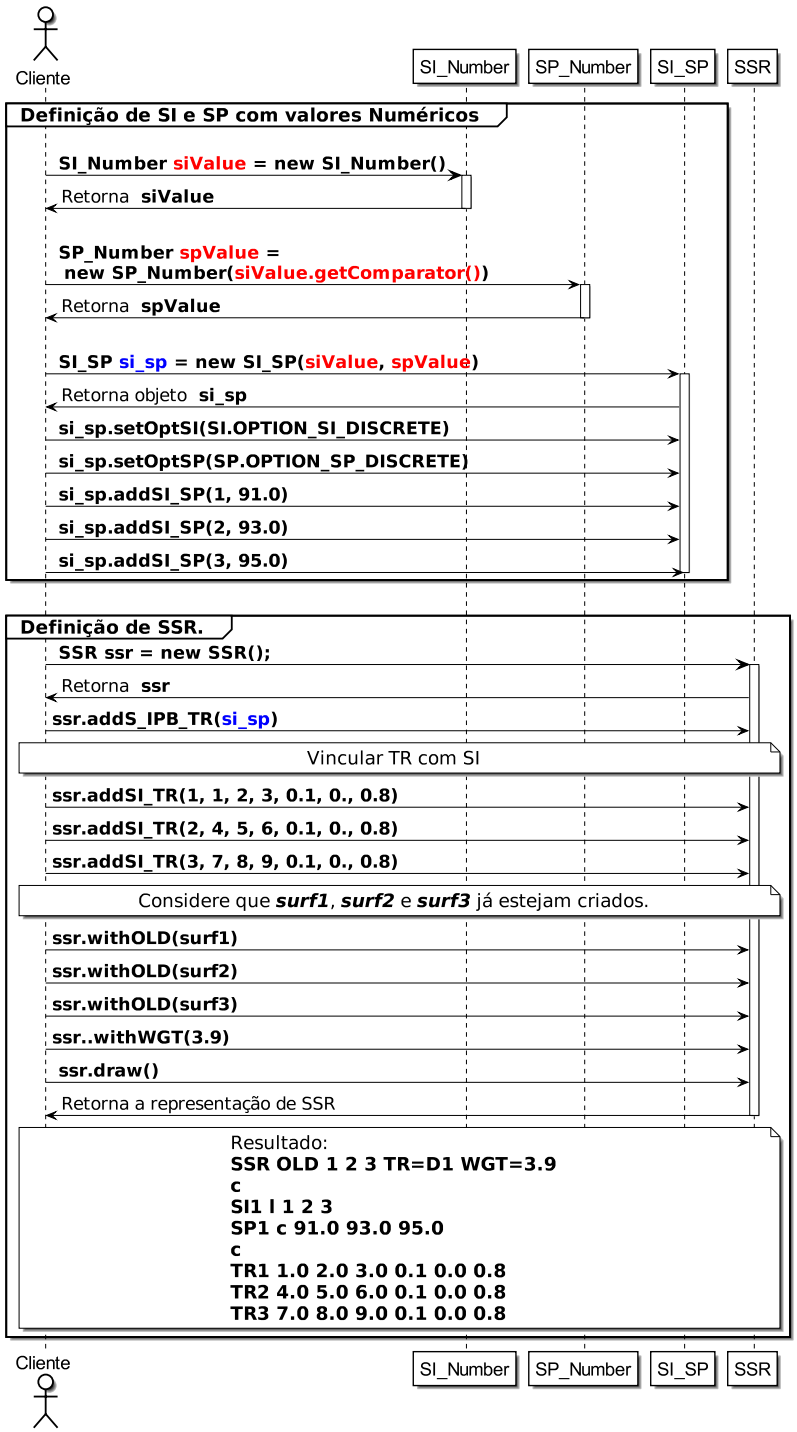

Figura C.49: Diagrama de sequencia referente à criação do cartão *SSR*.

Cartão Mode: O cartão *Mode* é utilizado para o *MCNP* saber qual ou quais partículas ele deve transportar. Sua sintaxe é "*Mode* mnemônicos_das_partículas ". A classe *Mode* somente suporta quatro tipos de partículas, fótons, elétron, nêutrons e prótons. Porém ela pode ser facilmente estendida para as outras partículas.

Na Figura C.50 há o diagrama de classes refente ao cartão *Mode*.

A classe *Mode* somente possui quatro métodos a fim de adicionar as partículas que serão transportadas. O método *addPhoton* para adicionar fótons o método *addElectron* para adicionar elétrons o método *addNeutron* para adicionar nêutrons e o método *addProtons* para adicionar prótons.

Para extrair a representação do cartão *Mode* é utilizado o método *draw*.

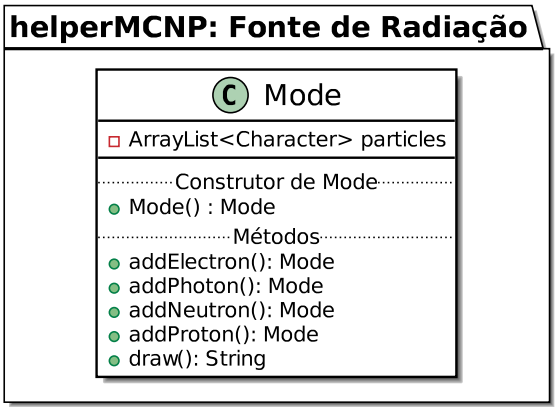

Figura C.50: Diagrama de classes referente ao cartão *Mode*.

Para exemplificar o uso da classe *Mode* é apresentado o diagrama de sequência na Figura C.51.

Os dois primeiros grupos adicionam somente um tipo de partícula a ser transportada. Já no último grupo são adicionados fótons, elétrons e nêutrons. A extração do resultado é feita mediante o método *draw*.

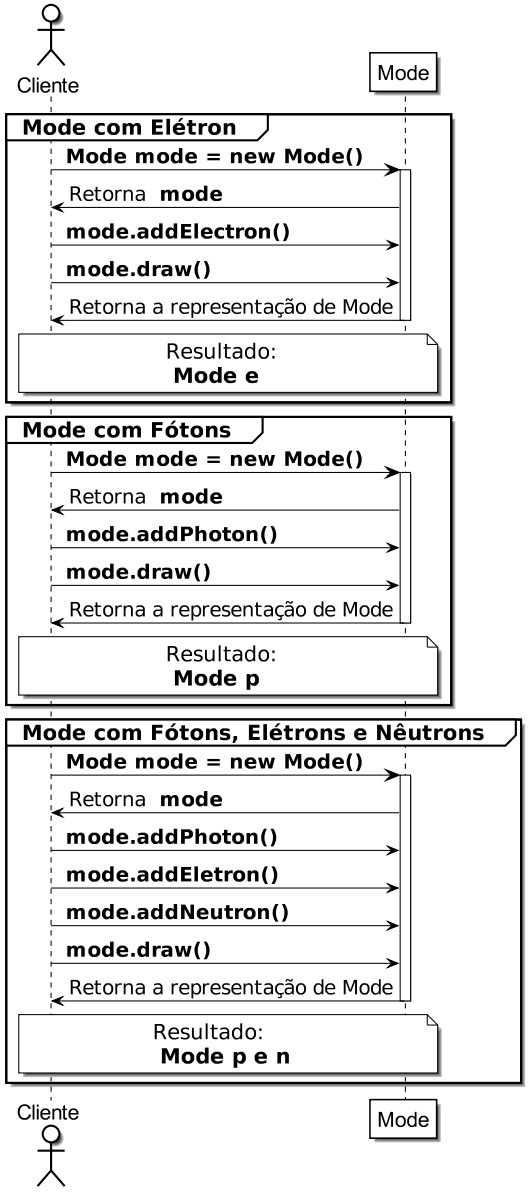

Figura C.51: Diagrama de sequência para exemplificar a criação do cartão *Mode*.

C.1.4 Construção dos Tally.

A seguir serão descritas as classes contidas em *helperMCNP* responsáveis por criar o *tally* para o *MCNP*.

Parâmetro de Tally DE e DF: Os parâmetros de *Tally DE* e *DF* permitem que seja declarada uma função de dose em função da energia a fim de modificar os valores de um *tally* regular.

A sintaxe para declarar esses dois cartões são: "*DEn a* e_1 ... e_k" e "*DFn b* f_1 ... f_k". A Tabela C.12 contem a descrição de cada parâmetro. As informações dessa tabela foram

retiradas do manual do *MCNP* (85).

Tabela C.12 - Descrição dos parâmetros dos cartões *DE* e *DF* que estão contidos em *helperMCNP*.

Parâmetros	Descrições
n	Número de identificação do parâmetro de *Tally DE_DF*.
a	São os modos como os valores de *DE* serão interpretados. Há duas opções.
	Se a=LOG, então ocorre uma interpolação logarítmica.
	Se a=LIN, então ocorre uma interpolação linear.
b	São os modos como os valores de *DF* serão interpretados. Há duas opções.
	Se b=LOG, então ocorre uma interpolação logarítmica.
	Se b=LIN, então ocorre uma interpolação linear.
e_i	São os valores de energia.
f_i	São os valores da função de dose correspondentes à i-ésima energia.

A seguir, na Figura C.52, há o diagrama de classes de *DE_DF* o qual constrói os cartões *DE* e *DF*.

Há três formas de se construir uma instância da classe *DE_DF*. Na primeira não é passado nenhum parâmetro ao construtor o qual constrói um *DE_DF* com número de identificação igual a zero. Na segunda já é especificado o número de identificação. Na terceira e última além do número de identificação também é especificado um *Comparator* a fim de ordenar os valores de *DE* e *DF*, tendo como referência *DE*.

O número de identificação poderá ser modificado pelo método *withNumber* e os valores de *DE* e *DF* serão adicionados mediante o método *addDE_DF*.

O modo como os valores de *DE* e *DF* serão interpretados são adicionados através dos métodos *setOptionDE* e *setOptionDF*. Para extrair a representação de *DE_DF* é utilizado o método *draw*.

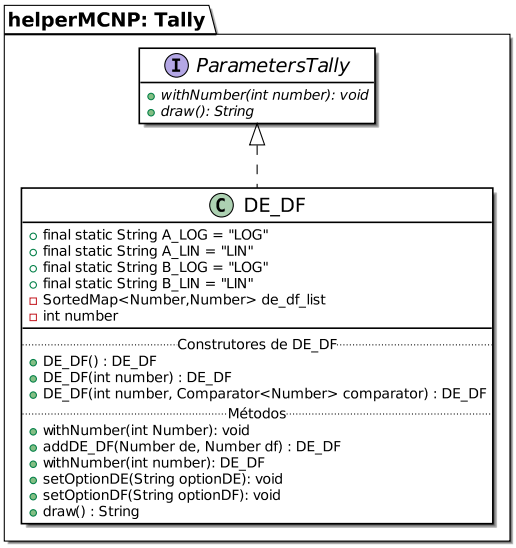

Figura C.52: Diagrama de classes de *DE_DF*.

Na Figura C.53 há um exemplo do uso da classe *DE_DF* mediante um diagrama de sequência.

Primeiramente é criado o objeto **de_df**, da classe *DE_DF*, com número de identificação igual à 8. A seguir são adicionados os modos como os dados de *DE* e *DF* serão interpretados. Depois os dados são adicionados e é requerido, por fim, a representação de *DE_DF* pelo método *draw*.

Se não for especificado nenhum *Comparator* então a ordenação é feita em ordem crescente com base nos valores de *DE*.

Figura C.53: Diagrama de sequência para exemplificar o uso da classe *DE_ DF*.

Parâmetro de Tally FM: O *FM* impõem um fator multiplicativo, que são seções de choque de aproximadamente todas as taxas de reações, aos valores de um *tally*.

Sua sintaxe é : "*FMn (bin set 1) (bin set 2) ... [T] [C]*". A Tabela C.13 contem as descrições dos parâmetros desse parâmetro de *Tally* que foram implementadas no *helperMCNP*. As informações dessa tabela foram retiradas do manual do *MCNP* (85).

Tabela C.13 - Descrição dos parâmetros de *FM* que estão contidos em *helperMCNP*.

Parâmetros	Descrições
n	Número de identificação do *Tally FM*.
atenuador	*Flag* utilizada para indicar que é um atenuador ao invés de um conjunto de multiplicadores
m	Números de identificações dos materiais do cartão *Mm*. Se valor for -0 é considerado todos os materiais.
lista de reações	Lista dos números de reações especificadas pelo *MCNP* (85).

Para elaborar o *Tally FM*, pela biblioteca *helperMCNP*, foi criada a classe *FM*. Na Figura C.54 há o diagrama de classes de *FM*.

O construtor de *FM* recebe o número de identificação do *Tally FM*. Para inserir um material é utilizado o método *withMaterial* e no caso de requerer todos os materiais é utilizado o método *withAllMaterial*.

Para acrescentar o atenuador é usado o método *withAttenuator*. A lista dos valores de reações é acrescentada através do método *withReactions*. Para extrair a representação de *FM* usa-se o método *draw*.

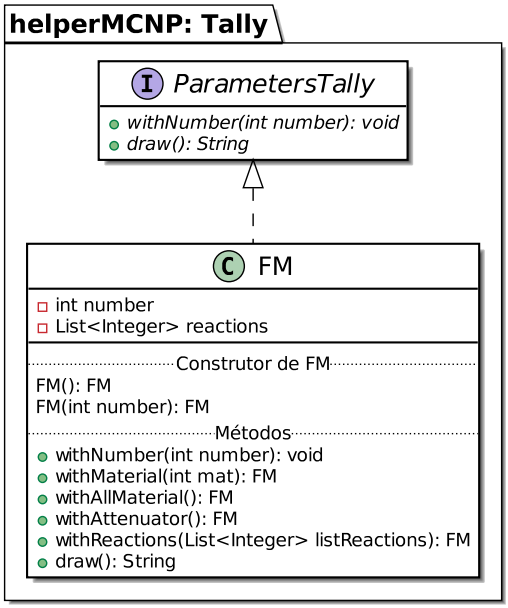

Figura C.54: Diagrama de classes de *FM*.

Na Figura C.55 é apresentado o diagrama de sequência que exemplifica o uso da classe *FM*.

Primeiramente é criado o objeto **fm**, da classe *FM*, com número de identificação igual à 4. A seguir são especificados todos os materiais, o atenuador e a lista de reações.

Para extrair a representação desse *Tally* foi utilizado o método *draw*.

Figura C.55: Diagrama de sequência para exemplificar o uso da classe *FM*.

Tally FMESH: O *Tally FMESH* defini um *tally* sobreposto à geometria do problema. Os resultados gerados por esse *tally* são escritos em um arquivo chamado *MESHTAL*. O *FMESH* faz uma estimativa do caminho do fluxo de partículas média em unidades de $partículas/cm^2$. Caso seja colocado um asterisco na frente do *FMESH* faz essa mesma estimativa porém em MeV/cm^2.

Apresenta a sintaxe: "*FMESHn:<p1> parâmetro(s)=valore(s) ...* ". Na Tabela C.14 há todas as descrições dos parâmetros referente ao *Tally FMESH* que foram incorporadas na biblioteca *helperMCNP*. As informações dessa tabela foram retiradas do manual do *MCNP* (85).

Tabela C.14 - Descrição dos parâmetros do *Tally FMESH* que estão contidos em *helperMCNP*.

Parâmetros	Descrições
n	Número de identificação do *Tally FMESH* terminando com 4.
<p1>	Mnemônicos das partículas que serão consideradas. Somente pode ser "*N*"(Nêutron) , "*P*" (Fótons) ou "*E*" (Elétrons).
GEOM	Especifica qual será a geometria do *MESH*. Pode ser tanto Cartesiana (XYZ ou REC) ou Cilíndrica (RZT ou CYL).
ORIGIN	Coordenada X, Y e Z que corresponde à origem do *MESH*. Essa posição é a do centro inferior para coordenadas cilíndricas e inferior esquerda, da parte de trás, para coordenadas retangulares.

Continuação da Tabela C.14

Parâmetros	Descrições
OUT	Define como o resultado será apresentado. Se OUT=COL o resultado é gerado como uma lista de valores em forma coluna. Há o erro relativo associado a cada valor. Se OUT=CF é o mesmo formato que o do "COL" porém há a adição dos resultados do *Tally* multiplicados pelo volume. Se OUT=IJ, IK ou JK os resultados são impressos em forma de matriz. Dependendo do sistema escolhido tem-se: I=x ou r, J=y ou z e K=z ou θ. Se OUT=NONE não é gerado o arquivo *MESHTAL*
IMESH	Localização dos pontos que definem os limites da malha na direção x ou r.
IINTS	Número de pontos dentro da malha de *IMESH* na direção x ou r.
JMESH	Localização dos pontos que definem os limites da malha na direção y ou z.
JINTS	Número de pontos dentro da malha de *JMESH* na direção y ou z.
KMESH	Localização dos pontos que definem os limites da malha na direção z ou θ.
KINTS	Número de pontos dentro da malha de *KMESH* na direção z ou θ.

Para criar o *Tally FMESH* foi implementada a classe *FMESH* na biblioteca *helperMCNP*. Na Figura C.56 há o diagrama de classes de *FMESH*.

Cada parâmetro de entrada do *Tally FMESH* possui um método correspondente na classe *FMESH*. Para inserir algum parâmetro de *Tally* seja *DE_DF* ou *FM* há o método *withParams*.

Para extrair a representação de *FMESH* há o método *draw*.

A fim de exemplificar o uso da classe *FMESH* foi elaborado o diagrama de sequência apresentado na Figura C.57.

Antes de criar uma instância de *FMESH* foi criada a instância de *DE_DF* que é um parâmetro do *Tally*. Depois desse processo foi criado o objeto **fmesh**.

Nos passos seguintes são adicionados os parâmetros de entrada do objeto **fmesh** e por fim é adcionado o parâmetro do *Tally* **de_df**. A extração do resultado gerado é feita pelo método *draw*.

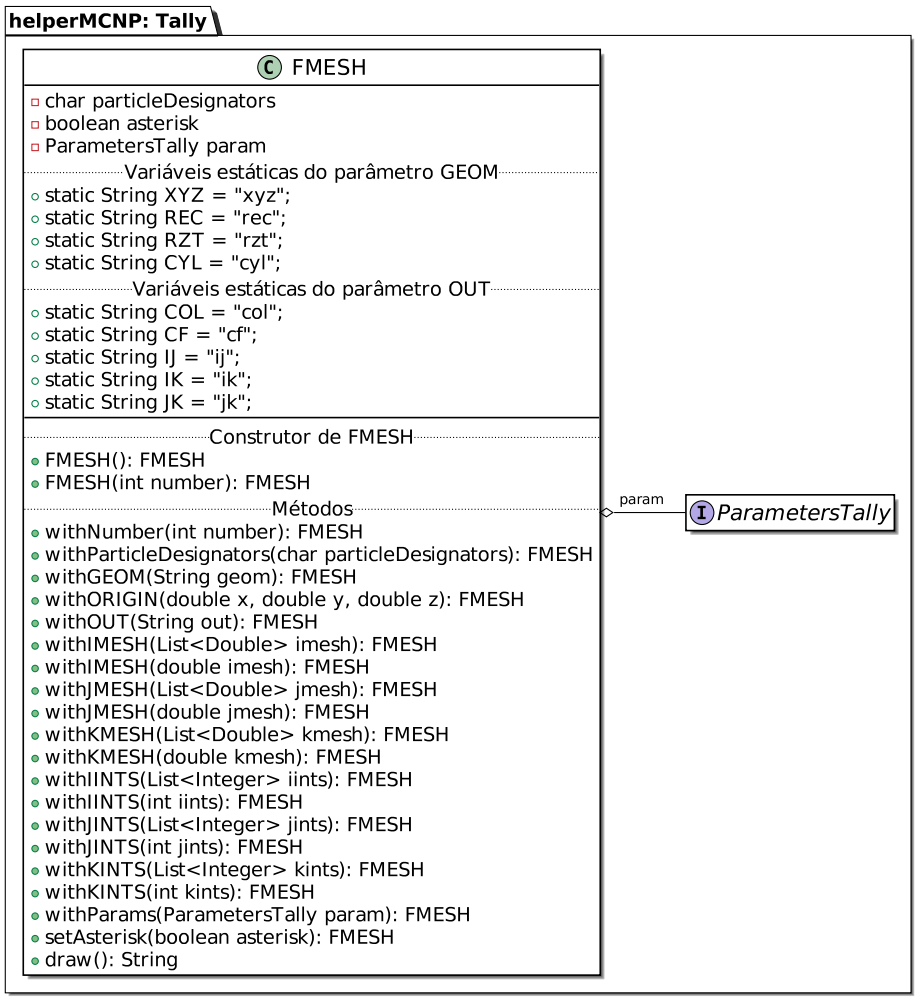

Figura C.56: Diagrama de classes de *FMESH*.

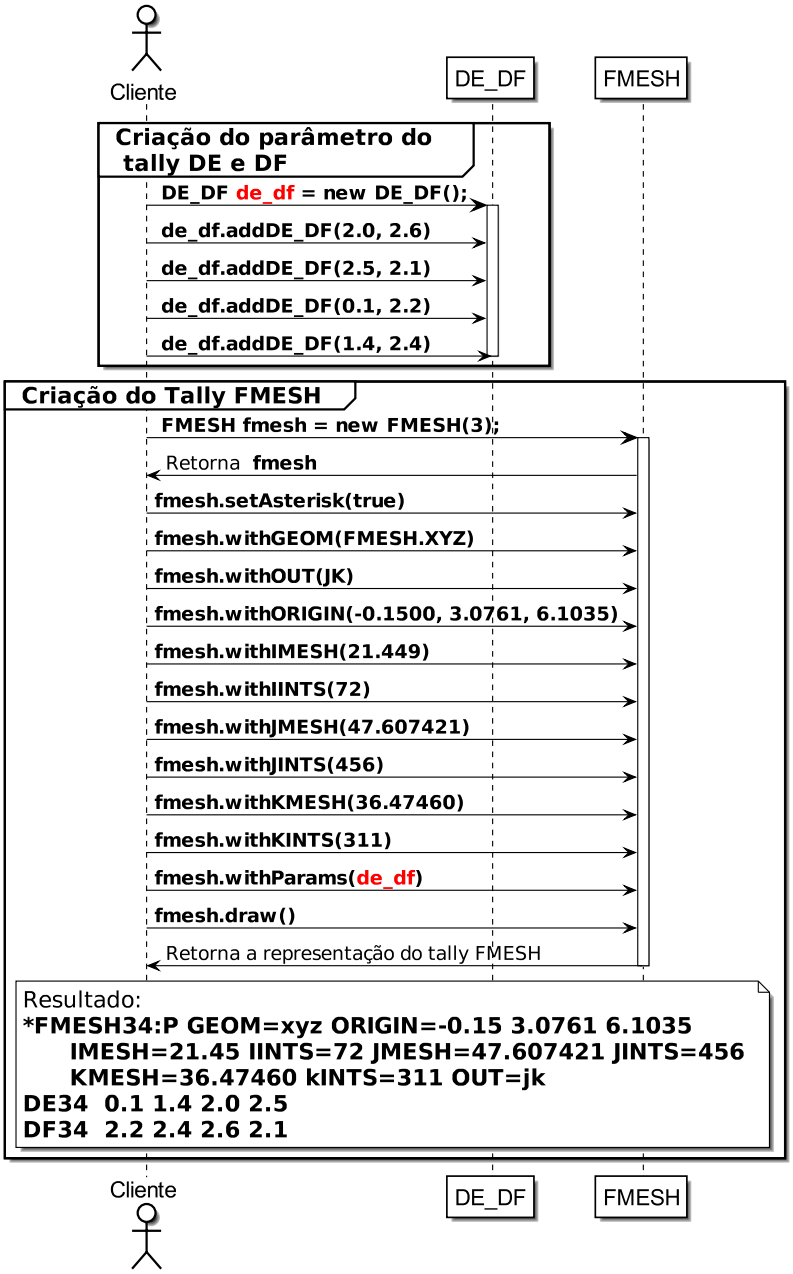

Figura C.57: Diagrama de sequência para exemplificar o uso da classe *FMESH*.

Tally F8: O *Tally F8* calcula a distribuição de energia em *pulsos* numa determinada célula. Se utilizado com asterisco, **F8*, calcula a energia depositada em MeV.

Sua sintaxe a qual foi implementada em *helperMCNP* foi "*[*]Fn:<p1> ((*s_i *...* s_i *<* c_1 *[*i_1*:*i_2 i_3*:*i_4 i_5*:*i_6*]))*". Na Tabela C.15 são apresentados as descrições desses parâmetros. As informações dessa tabela foram retiradas do manual do *MCNP* (85).

Tabela C.15 - Descrição dos parâmetros do *Tally F8* que estão contidos em *helperMCNP*.

Parâmetros	Descrições
*	*Opcional*. Ao invés de fornecer o resultado em *pulse* fornece em MeV.
n	Número de identificação do *tally* terminado com **8**.
<p1>	São os mnemônicos das partículas que serão consideradas no *tally*.
s_j	Número das células que estão dentro do *lattice*.
c_1	Número da célula do *lattice*.
i_i	Indica os intervalos em relação aos elementos do *lattice* e é baseado no cartão *FILL*.

Para criar o *Tally F8* a biblioteca *helperMCNP* possui a classe *F8*. A seguir, na Figura C.58 há o diagrama de classes de *F8*.

Há duas formas de se criar uma instância dessa classe. A primeira é sem passar nenhum parâmetro ao construtor o que gera um *Tally F8* sem um número antes do 8. A outra maneira é passado um número o qual será colocado anterior ao 8. Para requerer o asterisco antes do *F* é utilizado o método *withAsterisk*.

Para adicionar uma partícula é utilizado o método *addParticles* o qual recebe um caractere correspondente à partícula desejada. Esses caracteres podem ser adquiridos através da classe *Constants*.

Para requerer o *tally* em uma região com *lattice* é utilizado o método *withLatticeCell* que recebe como primeiro parâmetro uma lista de inteiros os quais referenciam às células contidas na região desejado do *lattice*, o segundo parâmetro é o número da célula do *lattice* e por fim os demais números especificam os limites da região desejada.

Para extrair a representação do *F8* é utilizado o método *draw*.

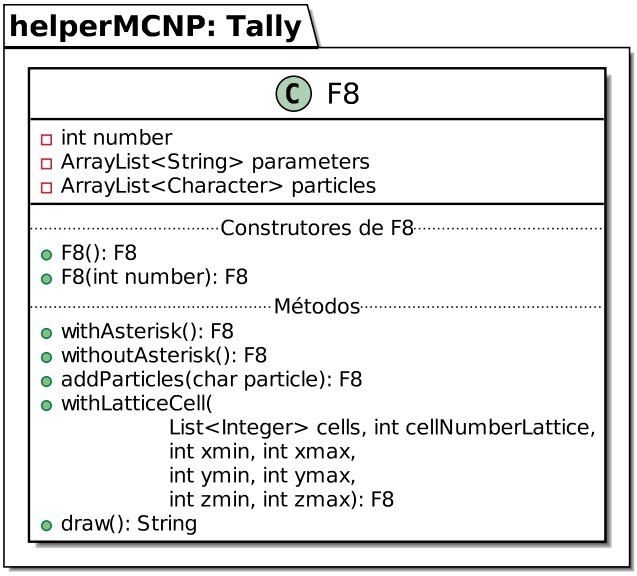

Figura C.58: Diagrama de classes correspondentes à classe *F8*.

Na Figura C.59 há um diagrama de sequência para exemplificar como é elaborado o *Tally F8* mediante a classe *F8* da biblioteca *helperMCNP*.

Primeiramente ocorre a criação do objeto **f8** com número adicional igual à 1. A seguir foram adicionados o asterisco, as partículas de elétrons e fótons e por fim foi especificado a região do *tally* mediante o método *withLatticeCell*.

A extração do *Tally F8* foi feita através do método *draw*.

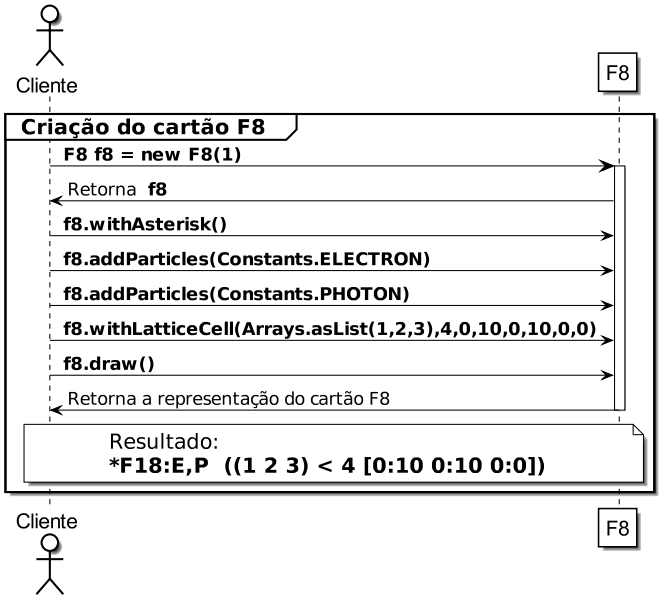

Figura C.59: Diagrama de sequência para exemplificar a criação do *Tally F8* pela biblioteca *helperMCNP*.

C.1.5 Definição dos Materiais.

A seguir será apresentado como os materiais envolvidos na simulação são cadastrados no arquivo de entrada do *MCNP*.

Cartão M: Os materiais que estão envolvidos na simulação são especificados através do cartão "*M*". Esse cartão apresenta a seguinte sintaxe, "$Mm\ zaid_1\ fraction_1\ ...\ zaid_n\ fraction_n$". Na Tabela C.16 são descritos os parâmetros da sintaxe do cartão *M*. As informações dessa tabela foram retiradas do manual do *MCNP* (85).

Tabela C.16 - Descrição dos parâmetros do Cartão *M*.

Parâmetros	Descrições
m	Número de identificação do material. Este número é o utilizado na célula.
$zaid_i$	Padrão para identificar o elemento químico. Sua sintaxe é "*ZZZAAA.abx*" ou "*ZZZAAA.*". *ZZZ* é o número atômico; Se AAA>0, então é o número de massa. Quando $AAA = 0$, então indica um elemento de ocorrência natural. "*ab*" é um identificador alfanumérico para uma biblioteca de elementos químicos. "*x*" é a classe do dado. A fim de representar um isótopo metaestável o valor de AAA é ajustado da seguinte forma: *AAA' = (AAA + 300) + (m * 100)*, em que m é o nível metaestável. O valor de m pode ser 1, 2, 3 ou 4.
$fraction_i$	Porcentagem do i-ésimo elemento de um material. Se *fraction*>0, então o valor é interpretado como fração atômica. Se *fraction*<0, então o valor é interpretado como fração em peso. Não pode haver casos em que há as duas opções.

A biblioteca *helperMCNP* possui duas classes para criar os materiais conforme a especificação do *MCNP*. Na Figura C.60 há o diagrama de classes, pertencentes à *helperMCNP*, que são destinadas à criação do cartão *M*.

A principal classe criadora dos materiais é a *MaterialMCNP*. Há cinco maneiras de se criar um objeto dessa classe. Veja a seguir.

1. O primeiro construtor da classe *MaterialMCNP* recebe três parâmetros. Nome do material, o número de identificação e um terceiro parâmetro o qual indica se as

frações dos elementos químicos devem ser interpretadas como frações atômicas ou de peso;

2. O segundo construtor da classe *MaterialMCNP* recebe dois parâmetros. Nome do material e o seu número de identificação. Nesse caso os elementos químicos são representados como fração em peso;

3. O terceiro construtor da classe *MaterialMCNP* recebe dois parâmetros. O número de identificação e o parâmetro que indica se as frações dos elementos químicos serão frações atômicas ou em peso. Nesse caso o nome do material será definido como "*Unnamed*";

4. O quarto construtor da classe *MaterialMCNP* recebe um único parâmetro que é o número de identificação. Nesse caso os padrões para o nome e o tipo de fração são os mesmos que os especificados no segundo e terceiro construtores;

5. O quinto e último construtor da classe *MaterialMCNP* não recebe nenhum parâmetro. Nesse caso o número de identificação é definido como zero e os demais parâmetros são padronizados conforme os construtores já especificados.

Ainda com base na Figura C.60, pode-se observar na classe *MaterialMCNP* que os parâmetros do nome do material e o número de identificação podem ser modificados depois que o objeto já estiver criado. Isso é feito, mediante os métodos, *withName* e *withNumber* respectivamente.

Também na classe *MaterialMCNP*, para adicionar os elementos químicos ao material é utilizado o método *addZaidFraction*. Esse método possui cinco formas de ser utilizado. Veja a seguir.

1. A primeira abordagem recebe cinco parâmetros. O primeiro e segundo parâmetros são o número atômico e de massa do elemento químico em questão, o terceiro é o nível metaestável, o quarto uma *String* que corresponde à uma determinada biblioteca "*abX*" e, por fim, o valor correspondente à fração do elemento químico. Com isso, é possível criar um elemento químico com nível metaestável e uma biblioteca "*abX*" escolhida;

2. A segunda abordagem recebe quatro parâmetros. Os números atômico e de massa , a *String* referente à uma biblioteca "*abX*" e a fração do elemento químico. Essa forma é utilizada quando não se deseja criar um elemento químico com nível metaestável;

3. A terceira abordagem recebe três parâmetros. Os números atômico e de massa e a fração do elemento químico. Assim, é possível criar um elemento químico, sem possuir nível metaestável, cuja biblioteca "*abX*" seja omitida e com isso o *MCNP* usa a biblioteca padrão;

4. A quarta abordagem recebe dois parâmetros que são o número atômico e a sua fração. Nesse caso, o número de massa é especificado como zero, o que indica a especificação de um elemento químico natural. Aqui também a biblioteca é omitida;

5. A quinta abordagem recebe três parâmetros. O número atômico, a fração desse elemento químico e a *String* referente à uma biblioteca "*abX*". Desse modo além de especificar um elemento natural é especificada a biblioteca "*abX*" a ser utilizada.

Para extrair a representação do cartão *M* conforme a especificação do *MCNP* é utilizado o método *draw*.

Na Figura C.60 há outra classe denominada *GroupOfMaterialMCNP*. Essa classe foi elaborada para auxiliar no armazenamento de um conjunto de objetos da classe *MaterialMCNP* e facilitar a extração da representação de todos esses objetos adicionados de maneira simples e rápida. Para adicionar um objeto da classe *MaterialMCNP*, dentro de *GroupOfMaterialMCNP*, é utilizado o método *addMaterial* e para extrair a representação de todos objetos adicionados é utilizado o método *draw*.

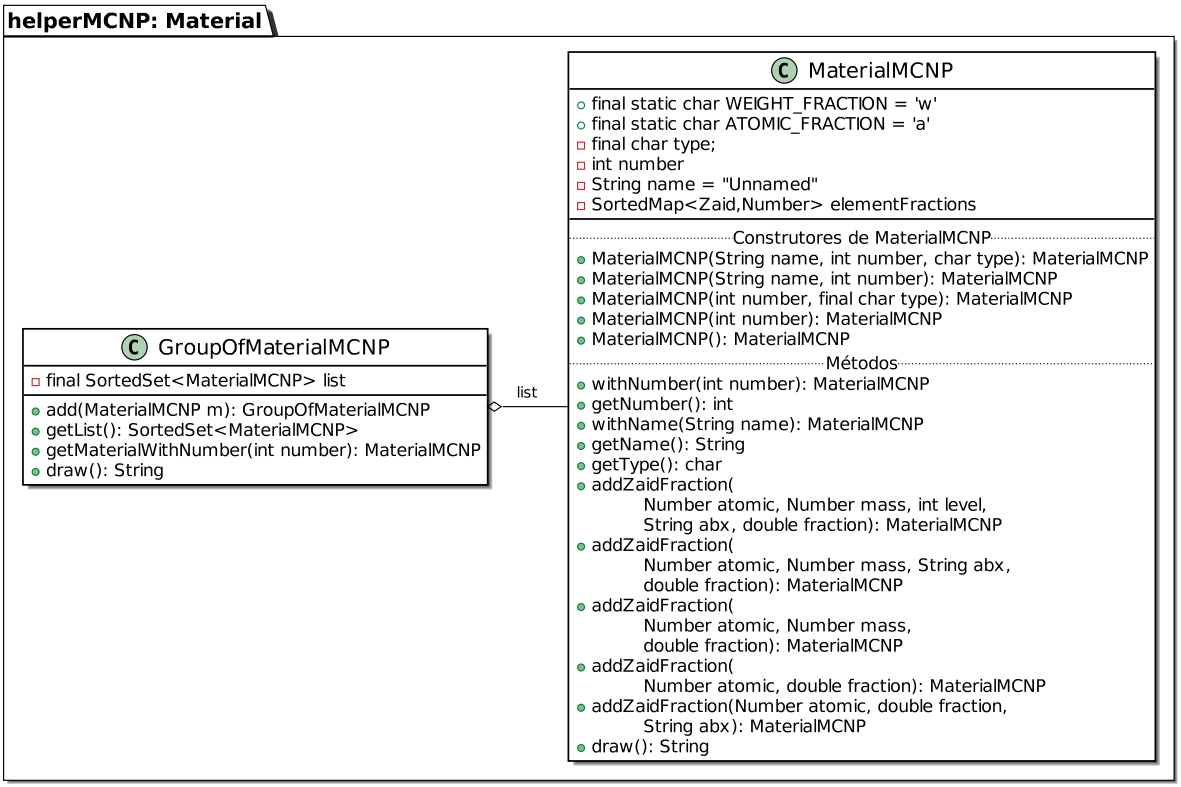

Figura C.60: Diagrama de Classes de *MaterialMCNP* e *GroupOfMaterialMCNP*.

Para exemplificar o uso da classe *materialMaterial* foi elaborado o diagrama de sequência presente na Figura C.61.

Neste diagrama, Figura C.61, foi representada a criação de um material referente à água. Primeiramente foi criado o objeto de *MaterialMCNP* passando ao construtor o nome do material,"*Água*", o número de identificação, dois, e a variável estática *ATOMIC_FRACTION* da classe *MaterialMCNP* o que significa que as frações de cada elemento serão consideradas frações atômicas.

Depois foram adicionados os elementos químicos Hidrogênio e Oxigênio ao o objeto criado, ***material***. Isso é feito através do método *addZaidFraction*, passando como parâmetro o número atômico, o número de identificação e a biblioteca "*abx*".

Por fim, mediante o método *draw*, foi extraída a representação desse material para o *MCNP*. Este resultado foi apresentada ao final do diagrama.

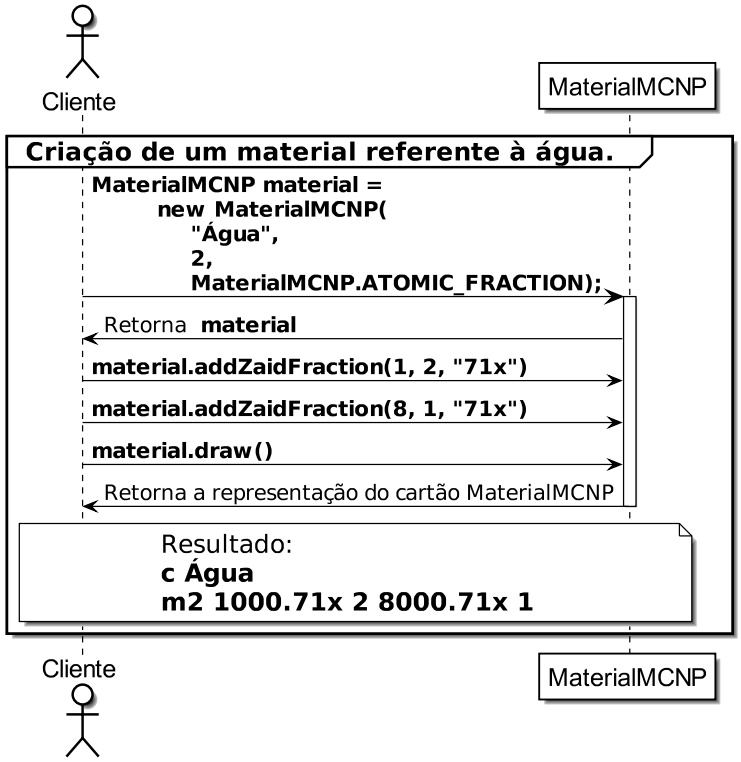

Figura C.61: Diagrama de sequência que exemplifica o uso da classe *MaterialMCNP*.

A seguir, na Figura C.62, é apresentado um exemplo, por meio de um diagrama de sequência, do uso da classe *GroupOfMaterialMCNP*.

Neste exemplo foram criados inicialmente dois objetos da classe *MaterialMCNP*, ***air*** e ***water***. Esses objetos foram criados de tal forma que as frações dos elementos químicos do objeto ***air*** representam frações em peso e as frações dos elementos químicos de ***water*** representam frações atômicas. Posteriormente, esses objetos foram acrescentados no objeto ***group*** da classe *GroupOfMaterialMCNP* através do método *addMaterial*.

Com isso, mediante um único método, *draw*, é possível requerer a representação de todos objetos de *MaterialMCNP* contidos em ***group***. Ao final do diagrama da Figura C.62 é apresentado o resultado gerado.

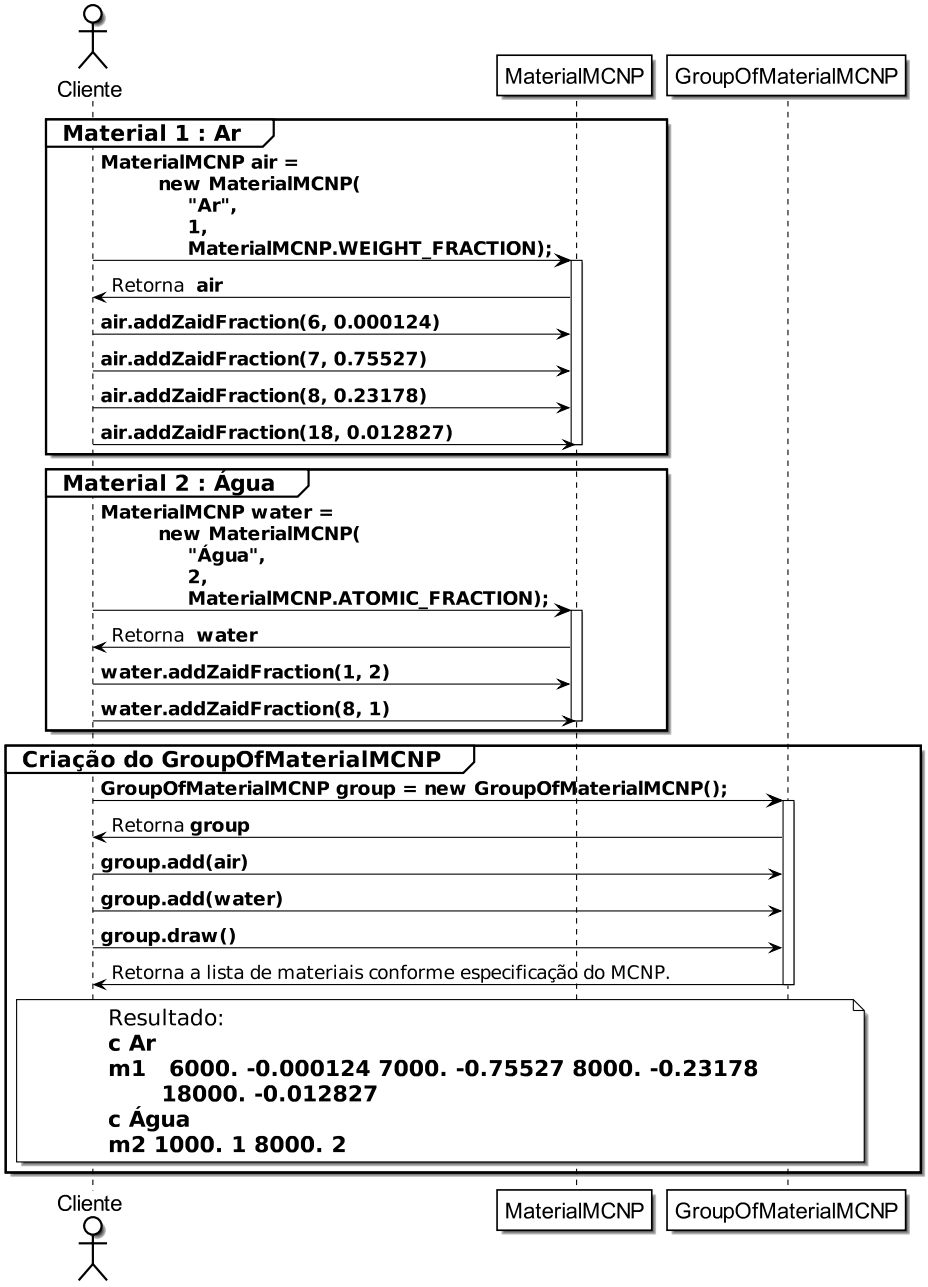

Figura C.62: Diagrama de sequência que exemplifica o uso da classe *GroupOfMaterialMCNP*.

C.1.6 Características da física da simulação.

Nesta seção será apresentada as classes, contidas em *helperMCNP*, envolvidas na física do problema e como essas informações são cadastradas no arquivo de entrada do *MCNP*.

Cartão PHYS: O cartão *PHYS*, do *MCNP*, é utilizado para especificar diversas opções referente à física de partículas. Essas opções são específicas para cada tipo de partícula. Na biblioteca *helperMCNP* foi implementado o cartão *PHYS* para dois tipos de partículas

que são fótons e elétrons.

Na Figura C.63 há o diagrama de classes correspondente ao cartão *PHYS*. A classe abstrata *Phys* é a superclasse das classes responsáveis pela física de partículas. Conforme Figura C.63, vê-se que tanto *PhysElectron*, classe responsável pela física dos elétrons, como *PhysPhoton*, classe responsável pela física dos fótons, estendem à classe *Phys*.

A classe *Phys* também estende a classe abstrata *ParameterInLine* que por sua vez é responsável por abstrair um padrão recorrente em alguns cartões do *MCNP*. Em alguns cartões, como é o caso do *PHYS*, seus parâmetros são colocados em sequência e cada um possui uma posição em específico em relação aos demais. Caso o usuário queira especificar que qualquer um desses parâmetros seja o padrão, então ele deverá colocar a letra "J" no lugar do parâmetro. E caso, o parâmetro que o usuário queira especificar como padrão for o último da lista, ele não precisa declará-lo, porém, se assim o quiser, deverá colocar a letra "J" no lugar do parâmetro. Nesse ponto, a classe *ParameterInLine* considera como padrão a opção de omitir o parâmetro, porém se o usuário quiser que estes últimos valores sejam substituídos por "J" então eles devem usar o método *withJafterLastParameterDeclared* passando “true” como parâmetro. Esta característica é herdada à todas as subclasses de *ParameterInLine*.

Na Figura C.63 é também possível observar que em cada subclasse concreta de *Phys*, como por exemplo em *PhysElectron*, há diversos métodos que são responsáveis por atribuir um valor aos seus respectivos parâmetros, por exemplo o método *withEmax* da classe *PhysElectron* o qual é responsável por armazenar o valor do parâmetro *emax* do cartão *PHYS* para elétrons, que será discutido posteriormente. É mediante esses métodos que os parâmetros serão inseridos nos objetos das subclasses de *Phys*.

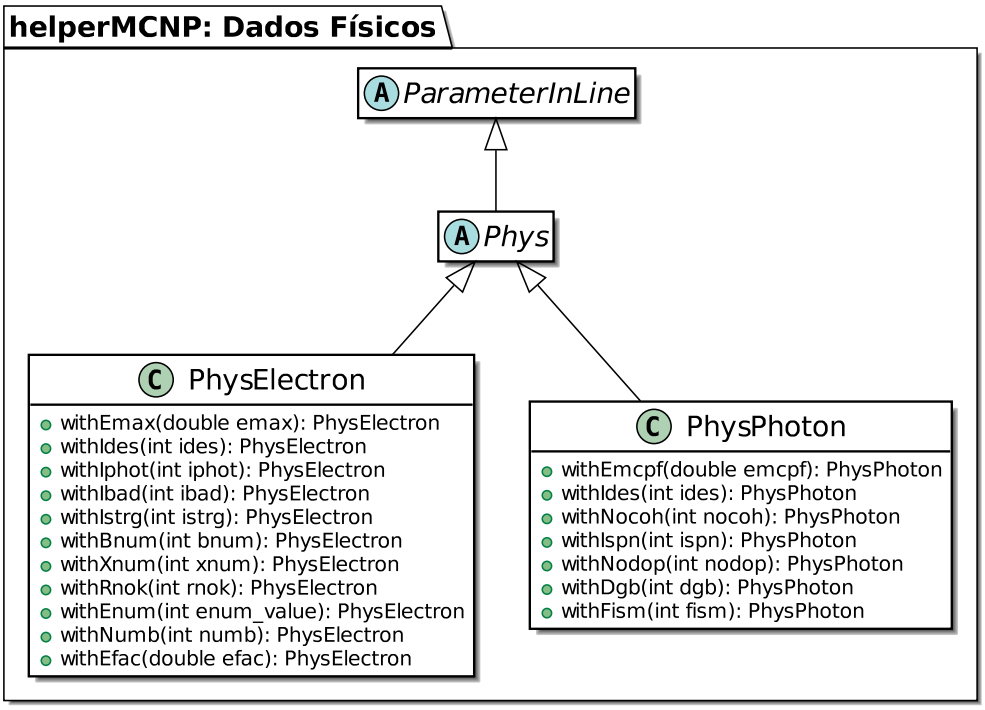

Figura C.63: Diagrama de classe de *PHYS* com suas subclasses *PhysPhoton* e *PhysElectron*.

A seguir é apresentada a classe *PhysPhoton* que é referente ao cartão *PHYS* para fótons. Ela é utilizada para definir alguns parâmetros em relação as interações dos fótons.

Sua sintaxe é: "*PHYS:P emcpf ides nocoh ispn nodop dgb fism*". Na Tabela C.17 há a descrição de cada uma desses parâmetros. As informações dessa tabela foram retiradas do manual do *MCNP* (85).

Tabela C.17 - Descrição dos parâmetros do cartão *PHYS* para fótons.

Parâmetros	Descrições
emcf	Valor de energia em *MeV* que será considerada como um limite superior para o detalhamento da física do fótons. A partir desse valor de energia o fóton será tratado com modelos físicos simplificados.
ides	Controla a geração de elétrons pela interação dos fótons quando é definido *MODE E*. Em casos que só são considerados fótons então ele controla a geração de fótons de *bremsstrahlung*. Caso *ides*=0, então a geração está ativada. (Padrão) Caso *ides*=1, então a geração está desativada.
nocoh	Controla o espalhamento coerente *Thomson*. Caso *nocoh*=0, então o espalhamento coerente é ativada. (Padrão) Caso *nocoh*=1, então o espalhamento coerente é desativada

Continuação da Tabela C.17

Parâmetros	Descrições
ispn	Controla a produção de partículas fotonuclear. Caso *ispn*=-1, então é análoga a produção de partículas fotonuclear. É amostrada uma interação de fóton por colisão. Caso *ispn*=0, então a produção de partículas fotonuclear é desligada (Padrão). Caso *ispn*=1, então a produção de partículas fotonuclear é tendenciosa. Um evento fotonuclear a cada evento foto atômico.
nodop	Controla a ampliação da Energia Doppler. Caso *nodop*=0, então a ampliação da Energia Doppler é ativada (Padrão). Caso *nodop*=1, então a ampliação da Energia Doppler é desativada.
dgb	Controla a produção de gama atrasada. Caso *dgb*=0, não cria gama atrasado. (Padrão) Caso *dgb*=-101, realiza a produção de gama atrasado com base nos dados do grupo-25 de emissão. Caso *dgb*=-102, realiza a produção gama atrasada com base nos dados da linha de emissão.
fism	Controla a seleção do método de foto fissão. caso *fism*=0, amostra a foto fissão pela biblioteca ACE. (Padrão) caso *fism*=1, amostra a foto fissão pelo modelo modelo LLNL. Para isso requer que *ispn* seja diferente de zero.

Na Figura C.64 há um diagrama de sequência que exemplifica o uso da classe *PhysPhoton* para criar o cartão *PHYS* para fótons. Neste exemplo foi criado um cartão *PHYS:P* que considera como padrão os parâmetros *Nocoh*, *Fism* e *Dgb*. Para especificar que um valor é padrão, não somente em relação à classe *PhysPhoton* mas à todas subclasses de *Phys*, basta não declará-lo. Isto é uma característica herdada da classe *ParameterInLine*. Dessa forma, os métodos responsáveis por declararem esses valores não foram utilizados, como é possível observar na Figura C.64.

Como dito anteriormente os valores que forem padrões e que estiverem intercalados à outros parâmetros declarados são substituídos pela letra "J". No diagrama de sequência da Figura C.64 esse processo é exemplificado pela omissão do parâmetro *Nocoh* em azul. Em relação aos parâmetros *Fism* e *Dgb*, representados em verde no diagrama da Figura C.64, eles são, por padrão, omitidos do resultado final. Porém, foi utilizado o método *withJafterLastParameterDeclared*, com o valor "*true*", e com isso a letra "J" aparece no resultado final para esses dois parâmetros.

Pode-se também observar, na Figura C.64, que embora o parâmetro *emcpf*, em vermelho, seja declarado depois de *ides* e *ispn*, ele aparece em primeiro da lista no resultado final. Por fim, para gerar o resultado final é utilizado o método *draw*.

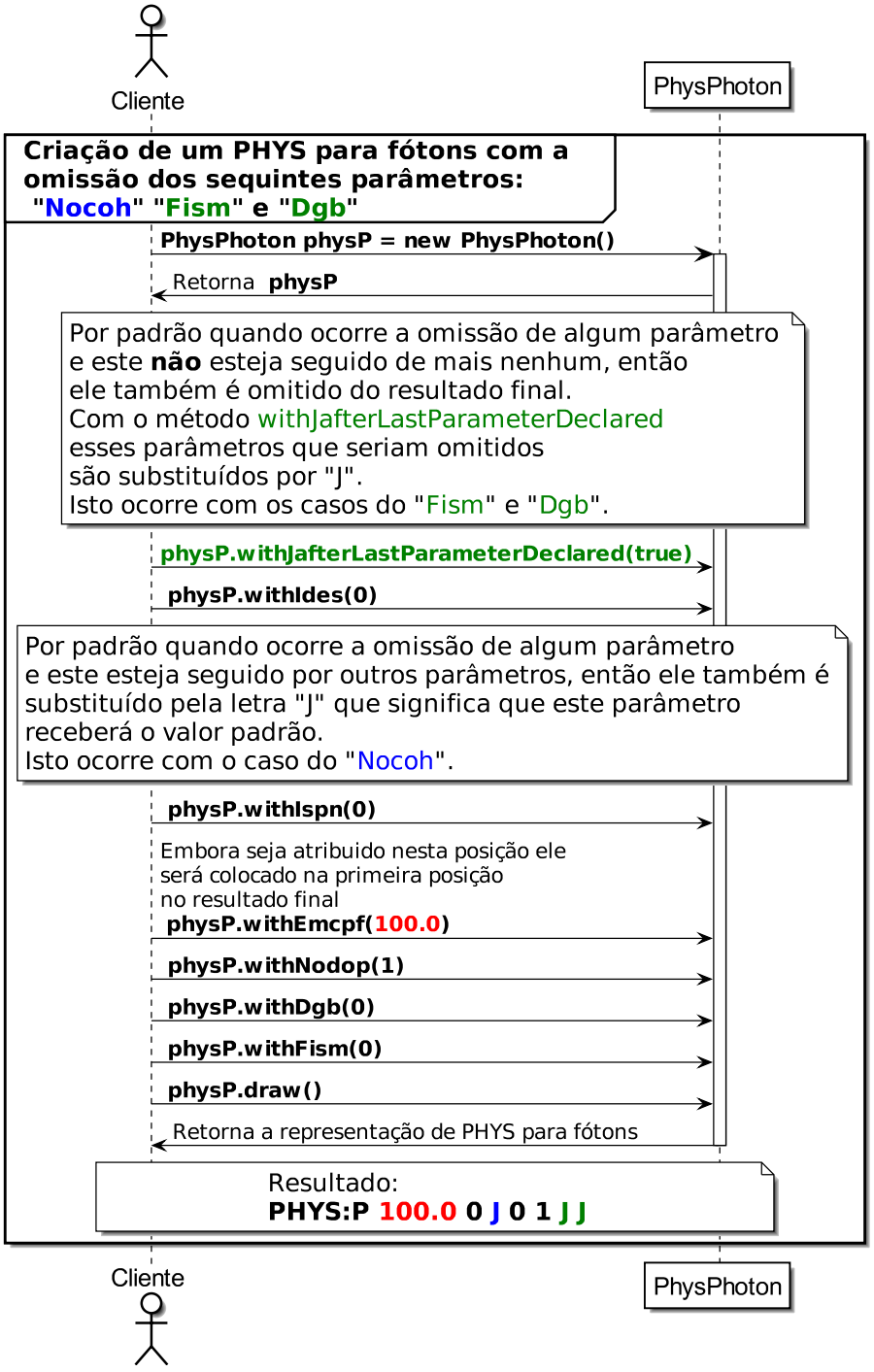

Figura C.64: Diagrama de sequência para criar o cartão *PHYS* para fótons através da classe *PhysPhoton*.

Em seguida é apresentada a classe *PhysElectron* que é referente ao cartão *PHYS* para elétrons do *MCNP*. Ela é utilizada para definir alguns parâmetros em relação as interações

dos elétrons.

Sua sintaxe é: "*PHYS:E emax ides iphot ibad istrg bnum xnum rnok enum numb efac*". Na Tabela C.18 há a descrição de cada uma desses parâmetros. As informações dessa tabela foram retiradas do manual do *MCNP* (85).

Tabela C.18 - Descrição dos parâmetros do cartão *PHYS* para elétrons.

Parâmetros	Descrições
emax	Energia máxima em MeV para os elétrons. Não serão simulados elétrons acima dessa energia. (Padrão é $100MeV$)
ides	Controla a geração de elétrons pela interação dos fótons quando é definido *MODE E*. Em casos que só são considerados fótons então ele controla a geração de fótons de *bremsstrahlung*. Caso *ides*=0, então a geração está ativada. (Padrão) Caso *ides*=1, então a geração está desativada.
iphot	Controla a produção fótons por elétrons. Caso *iphot*=0, então a produção de fótons é ativada. (Padrão) Caso *iphot*=1, então a produção de fótons é desativada.
ibad	Controla o método de distribuição angular de *bremsstrahlung*. Caso *idab*=0, realiza uma total distribuição angular tabular de *bremsstrahlung*. (Padrão) Caso *idab*=1, realiza uma simples distribuição angular de *bremsstrahlung*.
istrg	Controla o tratamento na perda energética do elétron. Caso *istrg*=0, utiliza o método de dispersão de valores amostrados para calcular a perda energética em cada colisão do elétron. (Padrão) Caso *istrg*=1, utiliza o método de dispersão de valores esperados para calcular a perda energética em cada colisão do elétron.
bnum	Controla a produção de fótons de *bremsstrahlung* criados pelos elétrons. Caso *bnum*=0, não serão produzidos fótons de *bremsstrahlung*. Caso *bnum*>0, produz *bnum* vezes o número de fótons de *bremsstrahlung*. (Padrão *bnum*=1) Caso *bnum*<0, utilizada somente para transporte de elétrons, produz $\|bnum\|$ vezes o número de fótons.
xnum	Controla a amostragem da produção de raio-x por elétrons. Caso *xnum*>0, produz *xnum* vezes o número de elétrons que induzem a formação de raios-x. (Padrão *xnum*=1) Caso *xnum*=0, não haverá produção de raios-x por elétrons.

Continuação da Tabela C.18

Parâmetros	Descrições
rnok	Controla a criação de elétrons *knock-on* produzidos pela interação de elétrons.
	Caso *rnok*>0, produz *rnok* vezes o número de elétrons *knock-on*. (Padrão *rnok*=1)
	Caso *rnok*>0, não são produzidos elétrons *knock-on*.
enum	Controla a geração de elétrons secundários induzidos por fótons.
	Caso *enum*>0, produz *enum* vezes o número de elétrons secundários induzidos por fótons. (Padrão *enum*=1)
	Caso *enum*=0, não serão produzidos elétrons secundários induzidos por fótons.
numb	Constrola a produção de *bremsstrahlung* à cada subconjunto de elétrons.
	Caso *numb*>0, produz *numb bremsstrahlung* em cada subconjunto.
	Caso *numb*=0, produção análoga a de *bremsstrahlung*. (Padrão)
efac	Controla o espaçamento da energia do poder de freamento. (Padrão *efac*=0.917)
	Restrição: $0.8 le efac le 0.99$

Na Figura C.65 há um diagrama de sequência que exemplifica o uso da classe *PhysElectron* para criar o cartão *PHYS* para elétrons. Neste exemplo foi criado um cartão *PHYS:E* que considera os parâmetros de *Iphot* e *Ibad* como padrões. Esses parâmetros estão em verde na Figura C.65. Sendo assim, os métodos responsáveis por declararem esses valores não foram utilizados, como é possível observar na Figura C.65.

Assim como em *PHYS* para fótons, alguns parâmetros, representados de cor vermelha na Figura C.65, foram declarados fora de ordem e no entanto eles apareceram no resultado final na ordem especificada na sintaxe de *PHYS* para elétrons. Por fim, para gerar o resultado final foi utilizado o método *draw*.

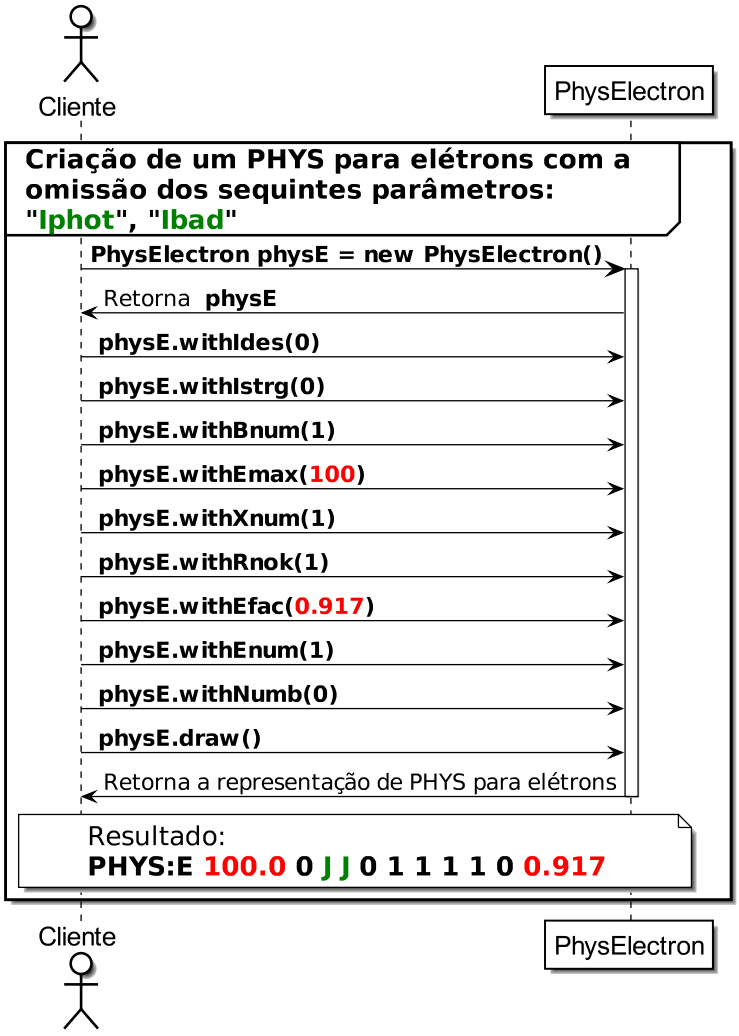

Figura C.65: Diagrama de sequência para criar o cartão *PHYS* para elétrons mediante a classe *PhysElectron*.

Cartão CUT: Esse cartão é utilizado para especificar determinados parâmetros de *cutoff* em relação à uma determinada partícula. Ele apresenta a seguinte sintaxe "*CUT:<p1> t e* WC_1 WC_2 *swtm*". Na Tabela C.19 há as descrições desses parâmetros. As informações dessa tabela foram retiradas do manual do *MCNP* (85).

Tabela C.19 - Descrição dos parâmetros do Cartão *CUT*.

Parâmetros	Descrições
<p1>	Designador referente à um determinada partícula.
t	Tempo de *cutoff*. A unidade é *shakes*, sendo que 1 *shake* equivale à 10^{-8}segundos.
e	Energia mínima de *cutoff* em MeV.
WC_1, WC_2	Pesos para os *cutoffs*.
swtm	Peso mínimo da fonte.

A fim de criar o cartão *CUT* a biblioteca *helperMCNP* possui a classe *CUT*. Na Figura C.66 há um diagrama de classes referente à classe *CUT*. Para criar um objeto da *CUT* apenas é necessário especificar a partícula desejada. Para isso, o construtor de *CUT* recebe um parâmetro "*char*" referente à partícula desejada. O usuário poderá encontrar esse valor na classe *Constants* de *helperMCNP*.

Para cada parâmetro que se deseja especificar há um método específico. Se o método referente a um determinado parâmetro não for requerido, o parâmetro daquele método será especificado com a letra "*J*" o que para o *MCNP* significa considerar um valor padrão.

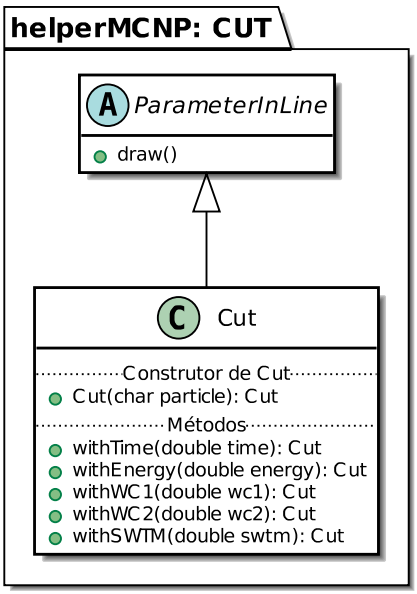

Figura C.66: Diagrama de classe correspondente ao cartão *CUT*.

A seguir, na Figura C.67, há um diagrama de sequência que exemplifica o uso da classe *CUT*, tendo como finalidade a construção do cartão *CUT* para o *MCNP*.

Nesse exemplo, foi criado o cartão *CUT* com os seguintes parâmetros. Designador da partícula referente ao elétron, tempo de *cutoff* de 10^8, peso WC_1 igual à -0.50, peso WC_2 igual à -0.25 e peso *swtm* igual à 0.50.

É importante mencionar que a ordem dos parâmetros importa ao cartão *CUT* e que a classe *CUT* abstrai essa informação. Isso pode ser observado, na Figura C.67, onde os métodos são chamados em uma ordem diferente à especificada na sintaxe do cartão, e mesmo assim o resultado final não foi afetado.

Por fim, a extração da representação desse cartão é feita pelo método *draw*.

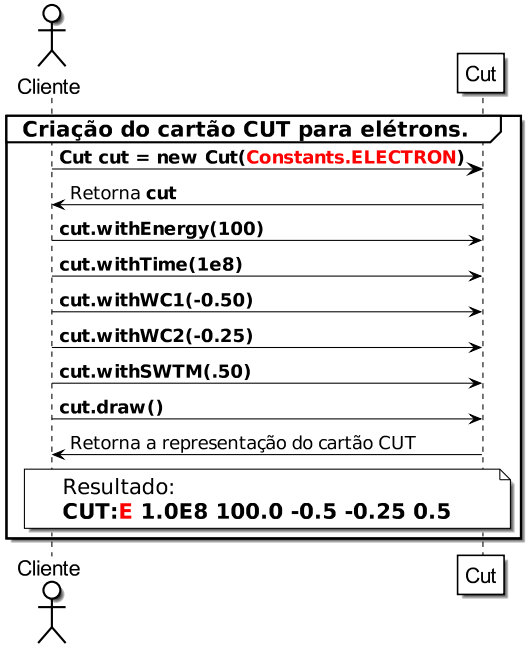

Figura C.67: Diagrama de sequência para exemplificar o uso da classe *CUT*.

C.1.7 Arquivo de saída do MCNP.

Durante esta seção, serão discutidas as classes, em *helperMCNP*, responsáveis por cadastrar as informações, no arquivo de entrada do *MCNP*, referente ao arquivo de saída do *MCNP*.

Cartão PRINT: Esse cartão é utilizado para gerar tabelas referentes aos resultados gerados na simulação. Ele apresenta a seguinte sintaxe "*PRINT* $[x_1\ x_2\ \ldots\ x_i\ \ldots]$". Os valores de x_i são números inteiros que representam um tabela de resposta do *MCNP*.

Foi implementada a classe *PRINT* na biblioteca *helperMCNP* para construir o cartão *PRINT*. Há diversas tabelas de resposta, porém a biblioteca só implementou duas que são a 30 e a 100. Caso haja necessidade, as demais tabelas podem ser implementadas facilmente.

Somente foram escolhidas estas duas tabelas pois são as mais recorrentes na área de Física Médica. A tabela 30 fornece a descrição dos *Tally* e a 100 fornece um tabela da seção de choque.

Na Figura C.68, há um diagrama de classes que corresponde à classe *PRINT*. Seu construtor não recebe nenhum parâmetro e há métodos específicos, na classe *PRINT*, para inclusão das tabelas. Esses métodos tem o padrão de receber um valor *booleano*. Caso se queira que uma tabela apareça é somente necessário utilizar o método referente à tabela

desejada e passar como parâmetro o valor *booleano* "*true*". Por padrão o valor *booleano* correspondente à cada tabela é "*false*".

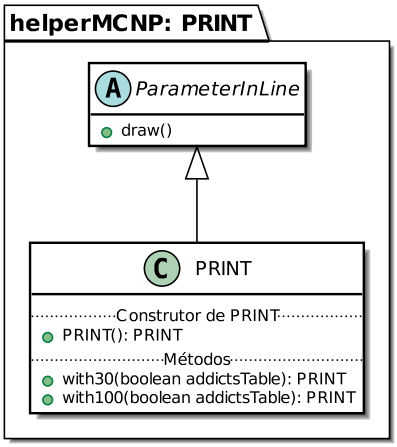

Figura C.68: Diagrama de classe correspondente ao cartão *PRINT*.

Para mostrar o uso da classe *PRINT* foi elaborado o diagrama de sequência apresentado na Figura C.69. Há dois exemplos nesse diagrama de sequência. O primeiro foi a elaboração de um cartão *PRINT* que mostrasse as tabelas 100 e 30. Já no segundo exemplo foi abordado construir um cartão *PRINT* somente com a tabela 30.

Para inclusão da tabela 100 foi utilizado o método *with100(true)* e para a inclusão da tabela 30 o método *with30(true)*. Por fim, para extrair a representação do cartão *PRINT* é utilizado o método *draw*.

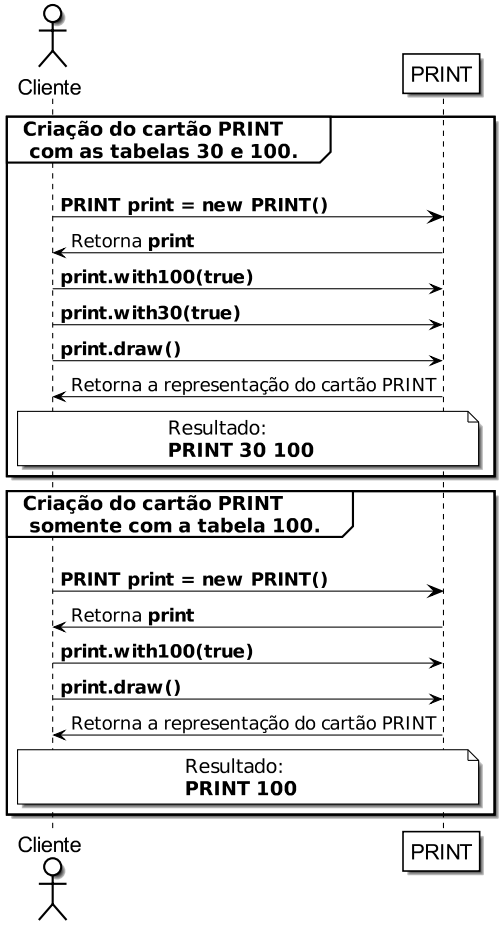

Figura C.69: Diagrama de sequência para exemplificar o uso da classe *PRINT*.

Cartão PRDMP: Esse cartão permite que o usuário controle o intervalo de tempo em que o arquivo de resposta é preenchido com as informações do *Tally*. Ele possui a seguinte sintaxe "*PRDMP ndp ndm mct ndmp dmmp*". Na Tabela C.20, há as descrições desses parâmetros. As informações dessa tabela foram retiradas do manual do *MCNP* (85).

Tabela C.20 - Descrição dos parâmetros do Cartão *PRDMP*.

Parâmetros	Descrições
ndp	Incremento para a impressão do *Tally*. Se *ndp*>0, então as respostas do *tally* são salvas no arquivo de saída do *MCNP* a cada intervalo de histórias igual a *ndp*. Se *ndp*<0, então as respostas do *tally* são salvas no arquivo de saída do *MCNP* a cada intervalo de tempo igual a *ndp*.

Continuação da Tabela C.20

Parâmetros	Descrições
ndm	Incremento para os *dumping* do arquivo RUNTPE. Se *ndm*>0, então os *dumping* são escritos no arquivo RUNTPE a cada intervalo de histórias igual a *ndm*. Se *ndm*<0, então os *dumping* são escritos no arquivo RUNTPE a cada intervalo de tempo igual a *ndm*.
mct	Controla a impressão do arquivo MCTAL. Caso *mct*>0, então escreve o arquivo MCTAL completo. Caso *mct*=0, então não escreve o arquivo MCTAL. Caso *mct*=-1, então escreve o arquivo MCTAL completo e junta outras informações, como nome do código, número da versão, identificador do problema, entre outros parâmetros. Caso *mct*=-2, então escreve o arquivo MCTAL completo e desliga a impressão do arquivo de saída padrão.
ndmp	Número máximo de *dumping* no arquivo RUNTPE.
dmmp	Controla a frequência do gráfico de flutuação do *tally*. Se *dmmp*<0, constrói o gráfico à cada 1000 partículas. Se *dmmp*=0, constrói o gráfico à cada 1000 partículas ou se for um processo à cada 10 vezes o número total de execuções. Se *dmmp*>0, constrói o gráfico à cada *dmmp* partículas. processo à cada 10 vezes o número total de execuções.

A biblioteca *helperMCNP* contêm a classe *PRDMP* que por sua vez é responsável por criar o cartão *PRDMP*. Na Figura C.70 há o seu diagrama de classes.

O construtor da classe de *PRDMP* não recebe nenhum parâmetro. Cada parâmetro do cartão *PRDMP* possui um método correspondente dentro da classe *PRDMP*. Caso um dos métodos seja omitido, seu valor é substituído pela letra "J" que indica ao *MCNP* a consideração de um valor padrão.

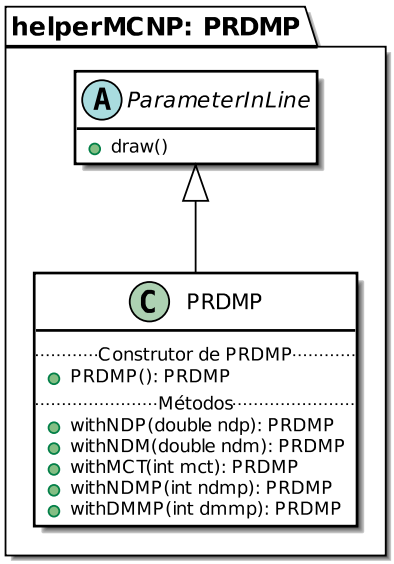

Figura C.70: Diagrama de classe correspondente ao cartão *PRDMP*.

O diagrama de sequência da Figura C.71 exemplifica o uso da classe *PRDMP*. Foi pretendido construir um cartão *PRDMP* que tivesse somente o parâmetro *NDM* igual á $1X10^8$.

Primeiramente foi criado um objeto da classe *PRDMP*, denominado **prdmp**. Mediante esse objeto foi requerido a inserção do parâmetro *NDM* mediante o método *withNDM*. Também houve uma chamada ao método *withJafterLastParameterDeclared*, que é da classe *ParameterInLine*, passando o valor *true*". Esse último método insere a letra "J" ao final do último parâmetro declarado.

A inclusão da letra "J" só é necessária se o parâmetro que se deseja considerar como padrão estiver entre dois outros parâmetros preenchidos. Dessa forma, colocar "J" no final não faz diferença no resultado final. A justificativa de criar esse método *withJafterLast-ParameterDeclared* se deve ao fato de muitos usuários gostarem dessa sintaxe, porém o comportamento padrão da biblioteca *helperMCNP* não é colocar "J" a partir do último parâmetro declarado.

Por fim, para extrair a representação de *PRDMP* é utilizado o método *draw*.

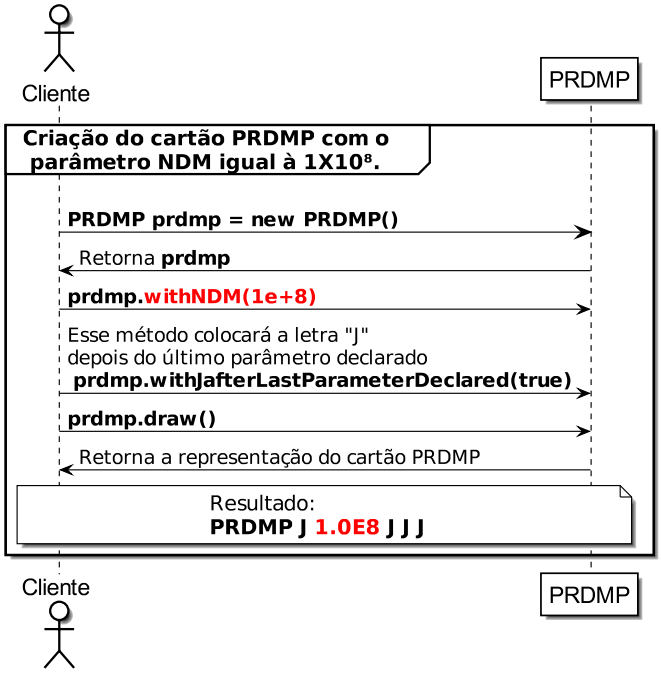

Figura C.71: Diagrama de sequência para exemplificar o uso da classe *PRDMP*.

C.1.8 Depurar simulação.

Esta seção descreve a classe *DBCN*, contida em *helper MCNP*, que cadastra no arquivo de entrada do *MCNP* o modo como a simulação deve ser depurada.

Cartão DBCN: Esse cartão é responsável por definir parâmetros da depuração da simulação. Apresenta a seguinte sintaxe "*DBCN* X_1 X_2 ... ". Há cem tipos de parâmetros que podem ser especificados. A biblioteca *helperMCNP* possui a classe *DBCN* para construir esse cartão e, no momento, só há dois parâmetros implementados por essa classe que são o X_{18} e o X_{49}.

Quando parâmetro X_{18} é igual à zero o *MCNP* utiliza o algoritmo de indexação de energia padrão, conhecido como tratamento "*bin-centered*". Caso X_{18} igual à 1 o algoritmo utilizado para indexação da energia é o da Série Integrada de Tiger, conhecido como tratamento "*nearest group boundary*". Por fim, quando X_{18} igual à 2 o *MCNP* utiliza a detalhada lógica amostral dispersa de Landau, conhecida como "*energy- and step-specific*", sendo está considerada a padrão pelo *MCNP*.

O parâmetro X_{49} é utilizado quando se deseja que o *MCNP* não faça verificação em relação a geometria do problema. Para isso, é necessário utilizar qualquer valor maior de zero para esse parâmetro.

Na Figura C.72 há o diagrama de classe referente à *DBCN*. Essa classe não recebe nenhum parâmetro em seu construtor. Para cada parâmetro do cartão *DBCN* implementado na classe *DBCN* há um método específico.

Apesar de ser implementado somente esses dois parâmetros a inclusão dos demais pode ser feita de maneira fácil, com a construção dos métodos, dentro da classe *DBCN*, específicos à cada parâmetro.

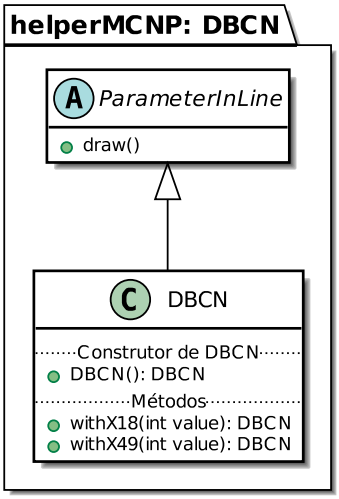

Figura C.72: Diagrama de classe correspondente ao cartão *DBCN*.

Na Figura C.73 há um diagrama de sequência que contêm um exemplo de como se utilizar a classe *DBCN*. O exemplo do diagrama é a construção do cartão *DBCN* com os parâmetros X_{18} igual à 1 e X_{49} igual à 2. Dessa forma, foi primeiramente criado o objeto, **dbcn**. o qual foi requisitado para a inclusão dos parâmetros X_{18} e X_{49} desejados. Nesse exemplo também foi utilizado o método *setUpCompressJ* com o valor booleano *true*. Esse método é da classe *ParameterInLine* e é utilizado para comprimir os "J" que aparecerão devido aos valores padrões dos outros parâmetros. Caso esse método não fosse utilizado iriam ocorrer no incio 17 letras “J”,em sequência, depois viria o valor de X_{18} seguido por mais trinta letras “J” e por fim o valor de X_{49}.

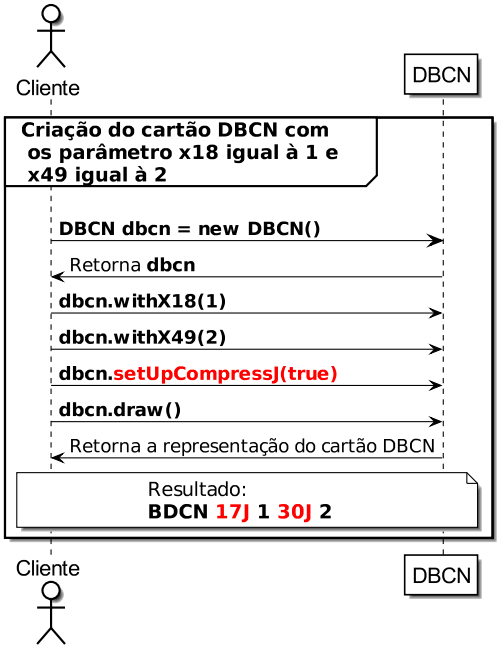

Figura C.73: Diagrama de sequência para exemplificar o uso da classe *DBCN*.

C.1.9 Finalização da simulação.

A seguir há a descrição da classe contida em *helperMCNP* responsável por criar o cartão que defini o termino da simulação.

Cartão NPS: Esse cartão indica ao *MCNP* que a partir de um determinado número de histórias ele deve encerrar a simulação. Isto se a simulação já tiver terminado por outra razão. Ele apresenta a sequinte sintaxe "*NPS npp npsmg*". O parâmetro "*npp*" é o número de histórias a ser executado no problema e o *npsmg* é o número de histórias considerando a fonte uma imagem radiográfica.

A Figura C.74 apresenta o diagrama de classe referente à classe *NPS*, responsável por construir o cartão *NPS*. O construtor de *NPS* não recebe nenhum parâmetro. O método *withNPP* especifica o valor do parâmetro *npp* e o método *withNpsmg* do parâmetro *npsmg*.

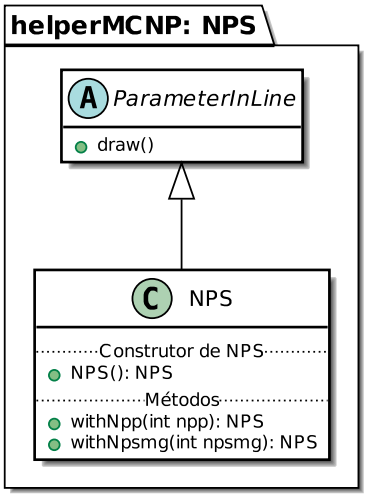

Figura C.74: Diagrama de classe correspondente ao cartão *NPS*.

Foi abordada a construção do cartão *NPS* com o parâmetro *npp* igual à 1000000. Primeiramente é criado o objeto, **nps**, da classe *NPS*. Depois o valor de *npp* é acrescentado com o método *withNpp*. Por fim, a extração da representação do cartão *NPS* é realizada através do método *draw*.

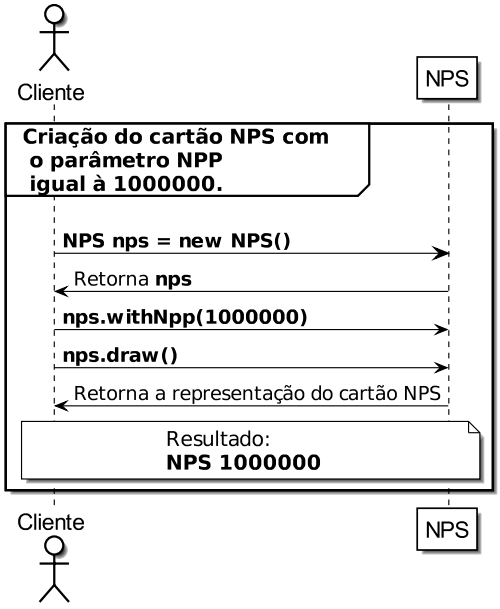

Figura C.75: Diagrama de sequência para exemplificar o uso da classe *NPS*.

Referências Bibliográficas

[1] SIANTAR, C. L. H.; MOSES, E. I. The peregrinetm program: using physics and computer simulation to improve radiation therapy for cancer. *European journal of physics*, IOP Publishing, v. 19, n. 6, p. 513, 1998.

[2] GIERGA, D.; YANCH, J.; SHEFER, R. *Therapeutic Applications of Monte Carlo Calculations in Nuclear Medicine*. [S.l.]: IOP Publishing, 2003. 324–347 p.

[3] HOGSTROM, K. R.; MILLS, M. D.; ALMOND, P. R. Electron beam dose calculations. *Physics in medicine and biology*, IOP Publishing, v. 26, n. 3, p. 445, 1981.

[4] RIVARD, M. J. et al. Update of aapm task group no. 43 report: A revised aapm protocol for brachytherapy dose calculations. *Medical physics*, American Association of Physicists in Medicine, v. 31, n. 3, p. 633–674, 2004.

[5] BOLCH, W. E. et al. Mird pamphlet no. 21: a generalized schema for radiopharmaceutical dosimetry—standardization of nomenclature. *Journal of Nuclear Medicine*, Soc Nuclear Med, v. 50, n. 3, p. 477–484, 2009.

[6] YORIYAZ, H.; STABIN, M. G.; SANTOS, A. dos. Monte carlo mcnp-4b–based absorbed dose distribution estimates for patient-specific dosimetry. *Journal of Nuclear Medicine*, Soc Nuclear Med, v. 42, n. 4, p. 662–669, 2001.

[7] YORIYAZ, H. et al. Physical models, cross sections, and numerical approximations used in mcnp and geant4 monte carlo codes for photon and electron absorbed fraction calculation. *Medical physics*, American Association of Physicists in Medicine, v. 36, n. 11, p. 5198–5213, 2009.

[8] MASSICANO, F. et al. Influence of elemental weight of human tissues estimated by ICCT software in absorbed dose calculation. *Revista Brasileira de Física Médica*, v. 4, n. 4, p. 1–5, 2011.

[9] JIANGA, H.; SECO, J.; PAGANETTIB, H. Effects of Hounsfield number conversion on CT based proton Monte Carlo dose calculations. *Medical Physics*, v. 34, n. 4, p. 1439–1449, 2008.

[10] PÖLZ, S. et al. Voxel2mcnp: a framework for modeling, simulation and evaluation of radiation transport scenarios for monte carlo codes. *Physics in medicine and biology*, IOP Publishing, v. 58, n. 16, p. 5381, 2013.

[11] ATTIX, F. H. *Introduction to radiological physics and radiation dosimetry*. [S.l.]: John Wiley & Sons, 2008.

[12] JOHNS, H.; CUNNINGHAM, J. The physics of radiology 4th edn (springfield, il: Thomas). 1983.

[13] BASKAR, R. et al. Cancer and radiation therapy: current advances and future directions. *International journal of medical sciences*, Ivyspring International Publisher, v. 9, n. 3, p. 193, 2012.

[14] SCAFF, L. A. *Física da Radioterapia: A Base Analógica de uma Era Digital*. [S.l.]: Projeto Sabe, São Paulo, 2010.

[15] PODGORSAK, E. B. Radiation oncology physics. *International Atomic Energy Agency*, v. 657, 2005.

[16] WILLIAMSON, J. F. Brachytherapy technology and physics practice since 1950: a half-century of progress. *Physics in medicine and biology*, IOP Publishing, v. 51, n. 13, p. R303, 2006.

[17] SAHA, G. B. *Fundamentals of nuclear pharmacy*. [S.l.]: Springer Science & Business Media, 2010.

[18] ZIESSMAN, H. A.; O'MALLEY, J. P.; THRALL, J. H. *Medicina nuclear*. [S.l.]: Elsevier Brasil, 2015.

[19] SANITÁRIA, B. M. da Saúde. Agência Nacional de V. Resolução rdc nº63, de 18 de janeiro de 2009. dispõe sobre as boas prática de fabricação de radiofármacos. 2002.

[20] DEWAN, N. et al. Diagnostic efficacy of pet-fdg imaging in solitary pulmonary nodules. potential role in evaluation and management. *CHEST Journal*, American College of Chest Physicians, v. 104, n. 4, p. 997–1002, 1993.

[21] AHMADIAN, A. et al. Quantitative interpretation of fdg pet/ct with myocardial perfusion imaging increases diagnostic information in the evaluation of cardiac sarcoidosis. *Journal of Nuclear Cardiology*, Springer, v. 21, n. 5, p. 925–939, 2014.

[22] SAH, B.-R. et al. Diagnostic performance of 18 f-fdg-pet/ct in vascular graft infections. *European Journal of Vascular and Endovascular Surgery*, Elsevier, v. 49, n. 4, p. 455–464, 2015.

[23] BAI, X. et al. Non-hodgkin lymphoma dominated by multiple organ extranodal disease revealed on fdg pet/ct. *Clinical nuclear medicine*, LWW, v. 40, n. 4, p. 360–363, 2015.

[24] BURTON, B. et al. Comparative analysis of f18-naf pet/ct and tc99m-mdp nuclear bone scans for detecting osseous metastases. *Journal of Nuclear Medicine*, Soc Nuclear Med, v. 56, n. supplement 3, p. 2724–2724, 2015.

[25] PURANIK, A. et al. Fdg pet/ct detects benign neurofibromas presenting as nodal masses: Imaging hallmarks of a diagnostic" red herring". *Indian Journal of Nuclear Medicine: IJNM*, Medknow Publications & Media Pvt. Ltd., v. 30, n. 2, p. 148, 2015.

[26] ELBOGA, U. et al. F-18 fdg pet/ct imaging in the diagnostic work-up of thyroid cancer patients with high serum thyroglobulin, negative i-131 whole body scan and suppressed thyrotropin: 8-year experience. *European review for medical and pharmacological sciences*, v. 19, n. 3, p. 396–401, 2015.

[27] KAEWPUT, C.; PUSUWAN, P. Severe hyponatremia: a comorbidity with i131 therapy in a patient with papillary thyroid cancer. *Journal of the Medical Association of Thailand= Chotmaihet thangphaet*, v. 97, n. 8, p. 886–890, 2014.

[28] BLESA, J. M. G. et al. Old and new insights in the treatment of thyroid carcinoma. *Journal of thyroid research*, Hindawi Publishing Corporation, v. 2010, 2010.

[29] MAHESHWARI, Y. K. et al. 131i therapy in differentiated thyroid carcinoma: Md anderson hospital experience. *Cancer*, Wiley Online Library, v. 47, n. 4, p. 664–671, 1981.

[30] BARBARO, D. et al. Radioiodine treatment with 30 mci after recombinant human thyrotropin stimulation in thyroid cancer: effectiveness for postsurgical remnants ablation and possible role of iodine content in l-thyroxine in the outcome of ablation. *The Journal of Clinical Endocrinology & Metabolism*, Endocrine Society, v. 88, n. 9, p. 4110–4115, 2003.

[31] PRESS, O. W. et al. Immunotherapy of non-hodgkin's lymphomas. *ASH Education Program Book*, American Society of Hematology, v. 2001, n. 1, p. 221–240, 2001.

[32] WALSH, G. *Biopharmaceuticals: biochemistry and biotechnology*. [S.l.]: John Wiley & Sons, 2013.

[33] JALILIAN A. R., K.-d. M. M. S. S. F.; MIRSADEGHI, L. Preparation and quality control of radiometal-dotarituximab. World Journal of Nuclear Medicine, v. 39, p. 135–141, 2008.

[34] RASANEH, S. et al. Radiolabeling of trastuzumab with 177 lu via dota, a new radiopharmaceutical for radioimmunotherapy of breast cancer. *Nuclear medicine and biology*, Elsevier, v. 36, n. 4, p. 363–369, 2009.

[35] PERSSON, M. et al. [177lu] pertuzumab: experimental studies on targeting of her-2 positive tumour cells. *European journal of nuclear medicine and molecular imaging*, Springer, v. 32, n. 12, p. 1457–1462, 2005.

[36] MILENIC, D. E. et al. Cetuximab: preclinical evaluation of a monoclonal antibody targeting egfr for radioimmunodiagnostic and radioimmunotherapeutic applications. *Cancer biotherapy & radiopharmaceuticals*, Mary Ann Liebert, Inc. 2 Madison Avenue Larchmont, NY 10538 USA, v. 23, n. 5, p. 619–632, 2008.

[37] VERA, D. R. B. et al. Preclinical evaluation of 177lu-nimotuzumab: A potential tool for radioimmunotherapy of epidermal growth factor receptor–overexpressing tumors. *Cancer biotherapy & radiopharmaceuticals*, Mary Ann Liebert, Inc. 140 Huguenot Street, 3rd Floor New Rochelle, NY 10801 USA, v. 26, n. 3, p. 287–297, 2011.

[38] POWSNER, R. A.; POWSNER, E. R. *Nuclear Medicine Physics*. [S.l.: s.n.], 2006. 206 p. ISBN 9781405104845.

[39] ZAIDI, H.; SGOUROS, G. *Therapeutic applications of Monte Carlo calculations in nuclear medicine*. [S.l.]: CRC Press, 2002.

[40] METCALFE, P.; KRON, T.; HOBAN, P. *The physics of radiotherapy x-rays and electrons*. [S.l.]: Medical Physics Publ., 2012.

[41] HIRONAKA, F. et al. *Medicina Nuclear–Princípios e Aplicações*. [S.l.]: São Paulo. Atheneu, 2012.

[42] STABIN, M.; XU, X. G. Basic principles in the radiation dosimetry of nuclear medicine. In: ELSEVIER. *Seminars in nuclear medicine*. [S.l.], 2014. v. 44, n. 3, p. 162–171.

[43] TSOUGOS, I. et al. Patient-specific internal radionuclide dosimetry. *Nuclear medicine communications*, v. 31, n. 2, p. 97–106, fev. 2010. ISSN 1473-5628. Disponível em: <http://www.ncbi.nlm.nih.gov/pubmed/20032767>.

[44] KOST, S. D. et al. Vida: A voxel-based dosimetry method for targeted radionuclide therapy using geant4. *Cancer Biotherapy and Radiopharmaceuticals*, Mary Ann Liebert, Inc. 140 Huguenot Street, 3rd Floor New Rochelle, NY 10801 USA, 2015.

[45] POSSANI, R. G. *Re-engenharia do software SCMS para uma linguagem orientada a objetos (Java) para uso em construções de phantoms segmentados*. Dissertação (Mestrado) — Instituto de Pesquisas Energéticas e Nucleares, Universidade de São Paulo, São Paulo, 2012.

[46] LEHMANN, J. et al. Monte carlo treatment planning for molecular targeted radiotherapy within the minerva system. *Physics in medicine and biology*, IOP Publishing, v. 50, n. 5, p. 947, 2005.

[47] JAN, S. et al. Gate: a simulation toolkit for pet and spect. *Physics in medicine and biology*, IOP Publishing, v. 49, n. 19, p. 4543, 2004.

[48] VISVIKIS, D. et al. Use of the gate monte carlo package for dosimetry applications. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, Elsevier, v. 569, n. 2, p. 335–340, 2006.

[49] SARRUT, D. et al. A review of the use and potential of the gate monte carlo simulation code for radiation therapy and dosimetry applications. *Medical physics*, American Association of Physicists in Medicine, v. 41, n. 6, p. 064301, 2014.

[50] CHIAVASSA, S. et al. Validation of a personalized dosimetric evaluation tool (oedipe) for targeted radiotherapy based on the monte carlo mcnpx code. *Physics in medicine and biology*, IOP Publishing, v. 51, n. 3, p. 601, 2006.

[51] GARDIN, I. et al. Voxeldose: a computer program for 3-d dose calculation in therapeutic nuclear medicine. *Cancer Biotherapy and Radiopharmaceuticals*, Mary Ann Liebert, Inc., v. 18, n. 1, p. 109–115, 2003.

[52] CLAIRAND, I. et al. Dose3d: Egs4 monte carlo code-based software for internal radionuclide dosimetry. *Journal of Nuclear Medicine*, THE SOCIETY OF NUCLEAR MEDICINE INC, v. 40, p. 1517–1523, 1999.

[53] GUY, M. J. et al. Rmdp: a dedicated package for 131i spect quantification, registration and patient-specific dosimetry. *Cancer Biotherapy and Radiopharmaceuticals*, Mary Ann Liebert, Inc., v. 18, n. 1, p. 61–69, 2003.

[54] REYNAERT, N. et al. Monte carlo treatment planning for photon and electron beams. *Radiation Physics and Chemistry*, Elsevier, v. 76, n. 4, p. 643–686, 2007.

[55] SECO, J.; VERHAEGEN, F. *Monte Carlo techniques in radiation therapy*. [S.l.]: CRC Press, 2013.

[56] RIVARD, M. J.; VENSELAAR, J. L.; BEAULIEU, L. The evolution of brachytherapy treatment planning. *Medical physics*, American Association of Physicists in Medicine, v. 36, n. 6, p. 2136–2153, 2009.

[57] FONSECA, G. P. et al. Dose specification for 192ir high dose rate brachytherapy in terms of dose-to-water-in-medium and dose-to-medium-in-medium. *Physics in medicine and biology*, IOP Publishing, v. 60, n. 11, p. 4565, 2015.

[58] LEE, C. Recent developments and best practice in brachytherapy treatment planning. *The British journal of radiology*, The British Institute of Radiology., v. 87, n. 1041, p. 20140146, 2014.

[59] CONSTANTINOU, C.; HARRINGTON, J. C.; DEWERD, L. A. An electron density calibration phantom for ct-based treatment planning computers. *Medical physics*, American Association of Physicists in Medicine, v. 19, n. 2, p. 325–327, 1992.

[60] PLESSIS, F. D. et al. The indirect use of ct numbers to establish material properties needed for monte carlo calculation of dose distributions in patients. *Medical physics*, American Association of Physicists in Medicine, v. 25, n. 7, p. 1195–1201, 1998.

[61] SCHNEIDER, W.; BORTFELD, T.; SCHLEGEL, W. Correlation between CT numbers and tissue parameters needed for Monte Carlo simulations of clinical dose distributions. *Physics in medicine and biology*, v. 45, n. 2, p. 4594–78, fev. 2000. ISSN 0031-9155.

[62] KAWRAKOW, I.; FIPPEL, M. Investigation of variance reduction techniques for monte carlo photon dose calculation using xvmc. *Physics in medicine and biology*, IOP Publishing, v. 45, n. 8, p. 2163, 2000.

[63] BONENFANT, É. et al. Fast gpu-based monte carlo simulations for ldr prostate brachytherapy. *Physics in medicine and biology*, IOP Publishing, v. 60, n. 13, p. 4973, 2015.

[64] FONSECA, G. P. et al. A medical image-based graphical platform—features, applications and relevance for brachytherapy. *Brachytherapy*, Elsevier, v. 13, n. 6, p. 632–639, 2014.

[65] AFSHARPOUR, H. et al. Algebra: Algorithm for the heterogeneous dosimetry based on geant4 for brachytherapy. *Physics in medicine and biology*, IOP Publishing, v. 57, n. 11, p. 3273, 2012.

[66] TAYLOR, R.; YEGIN, G.; ROGERS, D. Benchmarking brachydose: Voxel based egsnrc monte carlo calculations of tg-43 dosimetry parameters. *Medical physics*, American Association of Physicists in Medicine, v. 34, n. 2, p. 445–457, 2007.

[67] SIANTAR, C. H. et al. Description and dosimetric verification of the peregrine monte carlo dose calculation system for photon beams incident on a water phantom. *Medical physics*, American Association of Physicists in Medicine, v. 28, n. 7, p. 1322–1337, 2001.

[68] SEMPAU, J.; WILDERMAN, S. J.; BIELAJEW, A. F. Dpm, a fast, accurate monte carlo code optimized for photon and electron radiotherapy treatment planning dose calculations. *Physics in medicine and biology*, IOP Publishing, v. 45, n. 8, p. 2263, 2000.

[69] MA, C.-M. et al. Mcdose—a monte carlo dose calculation tool for radiation therapy treatment planning. In: *The Use of Computers in Radiation Therapy*. [S.l.]: Springer, 2000. p. 123–125.

[70] CHIBANI, O.; WILLIAMSON, J. F. Mcpi©: A sub-minute monte carlo dose calculation engine for prostate implants. *Medical physics*, American Association of Physicists in Medicine, v. 32, n. 12, p. 3688–3698, 2005.

[71] KAWRAKOW, I.; FIPPEL, M.; FRIEDRICH, K. 3d electron dose calculation using a voxel based monte carlo algorithm (vmc). *Medical physics*, American Association of Physicists in Medicine, v. 23, n. 4, p. 445–457, 1996.

[72] FIPPEL, M. Fast monte carlo dose calculation for photon beams based on the vmc electron algorithm. *Medical physics*, American Association of Physicists in Medicine, v. 26, n. 8, p. 1466–1475, 1999.

[73] KAWRAKOW, I.; FIPPEL, M. Vmc++, a fast mc algorithm for radiation treatment planning. In: *The Use of Computers in Radiation Therapy*. [S.l.]: Springer, 2000. p. 126–128.

[74] ZEE, W. Van der; HOGENBIRK, A.; MARCK, S. Van der. Orange: a monte carlo dose engine for radiotherapy. *Physics in medicine and biology*, IOP Publishing, v. 50, n. 4, p. 625, 2005.

[75] MARCK, S. C. van der; HOGENBIRK, A. Orange, a new, fast dose engine for radiotherapy treatment planning. *Radiation protection dosimetry*, NTP, v. 115, n. 1-4, p. 517–521, 2005.

[76] JAN, S. et al. Gate v6: a major enhancement of the gate simulation platform enabling modelling of ct and radiotherapy. *Physics in medicine and biology*, IOP Publishing, v. 56, n. 4, p. 881, 2011.

[77] PELOWITZ, D. B. et al. MCNPX^{TM} user's manual. *Los Alamos National Laboratory, Los Alamos*, 2005.

[78] CHIAVASSA, S. et al. Oedipe: a personalized dosimetric tool associating voxel-based models with mcnpx. *Cancer biotherapy & radiopharmaceuticals*, Mary Ann Liebert, Inc. 2 Madison Avenue Larchmont, NY 10538 USA, v. 20, n. 3, p. 325–332, 2005.

[79] GRUDZINSKI, J. et al. Patient specific treatment planning for systemically administered radiopharmaceuticals using pet/ct and monte carlo simulation. *Applied Radiation and Isotopes*, Elsevier, v. 68, n. 1, p. 59–65, 2010.

[80] MASSICANO, F. *Quantificação de imagens tomográfricas para o cálculo de dose em diagnose e terapia em medicina nuclear*. Dissertação (Mestrado) — Instituto de Pesquisas Energéticas e Nucleares, Universidade de São Paulo, São Paulo, 2010.

[81] RANDO® Phantoms. *THE PHANTOM LABORATORY PHANTOMS*. 2014. Disponível em: <http://www.phantomlab.com/products/rando.php>.

[82] DEITEL, H. M.; DEITEL, P. J.; SANTRY, S. E. *Advanced Java 2 platform How to program*. [S.l.]: Prentice Hall PTR, 2001.

[83] GAMMA, E. et al. *Design patterns: elements of reusable object-oriented software*. [S.l.]: Pearson Education, 1994.

[84] NIST. *National Institute of Standards and Technology*. 2014. Disponível em: <http://www.nist.gov/>.

[85] GOORLEY, T. et al. Initial mcnp6 release overview. *Nuclear Technology*, American Nuclear Society, v. 180, n. 3, p. 298–315, 2012.