UnB - UNIVERSIDADE DE BRASÍLIA

FGA - FACULDADE GAMA

PROGRAMA DE PÓS-GRADUAÇÃOEM ENGENHARIA BIOMÉDICA

RENDERIZAÇÃO DE ARQUÉTIPOS: UM ESTUDO DE CASO EM SUMÁRIO DE ALTA HOSPITALAR

ALFREDO MANOEL DE OLIVEIRA NETO

ORIENTADORA: Dra. Lourdes Mattos Brasil

DISSERTAÇÃO DE MESTRADO EM ENGENHARIA BIOMÉDICA

PUBLICAÇÃO: NUMERAÇÃO/2014

BRASÍLIA/DF: JULHO – 2014

UnB - UNIVERSIDADE DE BRASÍLIA

FGA - FACULDADE GAMA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA BIOMÉDICA

RENDERIZAÇÃO DE ARQUÉTIPOS: UM ESTUDO DE CASO EM SUMÁRIO DE ALTA HOSPITALAR

ALFREDO MANOEL DE OLIVEIRA NETO

DISSERTAÇÃO DE MESTRADO SUBMETIDA AO PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA BIOMÉDICA DA FACULDADE GAMA DA UNIVERSIDADE DE BRASÍLIA, COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM ENGENHARIA BIOMÉDICA.

APROVADA POR:

_____

Prof. Dra. Lourdes Mattos Brasil
(Orientadora)

_____

Prof. Dr. Marcelo Rodrigues Santos
(Examinador Externo)

_____

Prof. Dr. Fabiano Cavalcanti Fernandes
(Examinador Externo)

BRASÍLIA/DF, 29 DE JULHO DE 2014.

FICHA CATALOGRÁFICA

NETO, ALFREDO M. O.
Renderização de Arquétipos: Um Estudo de Caso em Sumário de Alta Hospitalar, [Distrito Federal] 2014.
99p.,210 x 297 mm (FGA/UnB Gama, Mestre, Engenharia Biomédica, 2014). Dissertação de Mestrado - Universidade de Brasília. Faculdade Gama. Programa de Pós-Graduação em Engenharia Biomédica.

1. Sumário de Alta Hospitalar
2. Arquétipos
3. Renderização
4. *Parser*

I. FGA UnB Gama/UnB.
II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

NETO, A. M. O. (2014). TÍTULO. Dissertação de Mestrado em Engenharia Biomédica, Publicação 024/2014, Programa de Pós-Graduação em Engenharia Biomédica, Faculdade Gama, Universidade de Brasília, Brasília, DF, 99p.

CESSÃO DE DIREITOS

AUTOR: Alfredo Manoel de Oliveira Neto.
TÍTULO: Renderização de Arquétipos: Um Estudo de Caso em Sumário de Alta Hospitalar.

GRAU: Mestre.

ANO: 2014.

É concedida à Universidade de Brasília permissão para reproduzir cópias desta dissertação de mestrado e para emprestar ou vender tais cópias somente para propósitos acadêmicos e científicos. O autor reserva outros direitos de publicação e nenhuma parte desta dissertação de mestrado pode ser reproduzida sem a autorização por escrito do autor.

Alfredo Manoel de Oliveira Neto
Qd 16 Casa 65 Setor Oeste - Gama.
CEP 72420-160 Brasília, DF – Brasil.

DEDICATÓRIA

Aos meus pais e minha filha.

AGRADECIMENTOS

À Deus pela capacitação, força e superação diária, na busca de mais um sonho.

A minha orientadora Profª. Lourdes Mattos Brasil e ao Profº. Edgard Costa Oliveira, pela paciência, estímulo no desenvolvimento do projeto, compreensão, incentivo e por acreditar e dar força em todos os momentos e etapas, também pela exigência e cobrança nas atividades, objetivando o aperfeiçoamento e qualidade do trabalho.

Aos professores, Georges Daniel AmvameNze, Suélia de Siqueira Rodrigues Fleury Rosa, Cristiano Jacques Miosso Rodrigues Mendes e Marília Miranda Forte Gomes, pelas explicações de conteúdo, incentivo e disponibilidade para ajudar.

Aos pesquisadores do projeto RES-BRASIL, especialmente ao Marcelo Rodrigues Santos, Jussara Macedo Rötzsch, Beatriz Faria Leão e Gabriela Alves, pelos esclarecimentos de conteúdos em momentos oportunos e incentivo.

Aos meus amigos que me auxiliaram em vários aspectos para o desenvolvimento das atividades do curso, sejam com palavras e até mesmo estudando junto. Em especial a, Marlete Maria, Roberto Aguiar, Juliana Elias, Daniel Souza, Henrique Maffon, Roozbeh Tahmasebi, Rafael Assunção, Alessandro Pinheiro e Luciana Rocha.

Aos meus pais pelas palavras de perseverança e compreensão e por acreditarem no meu potencial, e minha filha pelas tantas vezes que teve que me esperar estudar.

RESUMO

RENDERIZAÇÃO DE ARQUÉTIPOS: UM ESTUDO DE CASO EM SUMÁRIO DE ALTA HOSPITALAR

Autor: ALFREDO MANOEL DE OLIVEIRA NETO
Orientador: Prof. Dra. Lourdes Mattos Brasil
Programa de Pós-Graduação em Engenharia Biomédica
Brasília, 29 de Julho de 2014.

A comunicação entre homens e máquinas faz, cada vez mais, parte do dia a dia de milhões de pessoas, em especial para esse estudo de caso, os usuários de sistemas de informação em saúde. Este trabalho conta com as especificações da organização internacional openEHR, a qual possui uma especificação de um modelo de referência com informações pertinentes a todas especificidades e peculiaridades dos modelos de domínio em saúde, permitindo o desacoplamento das questões clínicas e tecnológicas, separando a semântica da informação e do conhecimento, bem como os conceitos clínicos do demográfico. A fundação openEHR representa essas informações na forma de arquétipos, que são modelos clínicos (validados, estruturados e codificados eletronicamente) livres para uso em qualquer sistema, e que podem ser reutilizados dentro de um conceito do domínio e em diversos cenários. O contexto estudado foi o Sumário de Alta Hospitalar, instrumento que trata da transição das informações do cuidado ao paciente, que foi realizado após a internação dando continuidade no tratamento em ambiente ambulatorial, também utilizado para orientação da comunicação entre profissionais do hospital e da atenção primária, devendo ser suficientes para garantir a segurança do paciente. As informações foram apresentadas de uma forma estruturada e organizada para que houvesse uma garantia de que os registros fossem apresentados de forma objetiva para compreensão do profissional que o receberá. Para representar essas informações, foi criado um protótipo de *software*, a fim de reproduzir em tela os tipos de objetos contidos nos arquétipos que representam o contexto do sumário de alta, visto que os arquétipos possuem um Modelo de Objetos que utilizam os conceitos de Orientação a Objetos (OO). Com base na análise dos resultados, a renderização é a representação gráfica em tela dos objetos extraídos dos arquétipos definidos em forma de *parser*.

Palavras-chaves: arquétipos, sumário de alta hospitalar, *parser* e protótipo.

ABSTRACT

RENDERING OF OPENEHR ARCHETYPES: A CASE STUDY IN SUMMARY OF DISCHARGE

Author: ALFREDO MANOEL DE OLIVEIRA NETO

Supervisor: Dr. Lourdes Mattos Brasil

Post-Graduation Program in Biomedical Engineering

Brasília, 29 July of 2014.

The communication between humans and machines has been part of everyday people life, particularly for this case of study, users of health information system. This study relies on the openEHR, which has specification of a reference model with relevant information to all specifics and particularities of the health domain models, enabling decoupling of clinical and technological issues, splitting out the semantics of information and knowledge, as well as the demographic clinical concepts. The openEHR Foundation represents this information in the form of archetypes, which are clinical models (validated, structured and coded electronically), free to use in any system, and can be reused within a domain concept in various scenarios. The context studied was the Summary of Hospital Discharge, instrument dealing with the transition of patient care information, which was carried out after the hospitalization continuing treatment in an outpatient environment, also used for guidance of communication between professionals in the hospital and primary care, and should be enough to ensure patient safety. The information was presented in a structured and organized way to guarantee that the records were presented objective manner for understanding of the professional who will receive. Representing this information, prototype *software* was created, in order to play back on screen the types of objects contained in the archetypes that represent the context of discharge summary, given that the archetypes have a model of objects using the concepts of Object Orientation (OO). Based on the analysis of the results, the rendering is the graphic representation on the screen of the archetypes extracted objects defined in form of *parser*.

Keywords: archetypes, summary of hospital discharge, parser and prototype

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE SÍMBOLOS, NOMENCLATURAS E ABREVIAÇÕES

ABNT – Associação Brasileira de Normas Técnica

ACM – *Association for ComputingMachinery*

ADL – *Archetype Definition Language*

AIM – *Advanced Informatics in Medicine*

AOM – *Archetype Object Model*

BD – Banco de Dados

BIREME – Biblioteca Regional de Medicina

BVS – Biblioteca Virtual em Saúde

cADL – *Constraint Archetype Definition Language*

CAPES – Coordenação de Aperfeiçoamento de Pessoal de Nível Superior

CEE-IS – Comissão de Estudo Especial de Informática e Saúde

CFM – Conselho Federal de Medicina

CKM – *Clinical Knowledge Manager*

CNS – Cartão Nacional de Saúde

dADL – *Data Archetype Definition Language*

DATASUS – Departamento de Informática do Sistema Único de Saúde

DCL – *Data Control Language*

DDL – *Data Definition Language*

DER – Diagrama de Entidade Relacionamento

DML – *Data Manipulation Language*

FGA – Faculdade Gama

FOPL – *First-Order Predicate Logic*

GEHR – *Good Electronic Health Record*

GUI – *Graphical User Interface*

GUJ – Grupo de Usuários Java

HTML - *Hyper Text Markup Language*

IBTI – Instituto Brasília de Tecnologia e Inovação

IEEE – *Institute of Electrical and Electronics Engineers*

IS – Informática em Saúde

ISO –*International Organization for Standardization*

ITA –Instituto Tecnológico Aeronáutico

JDK – *Java Development Kit*

JVM – *Java Virtual Machine*

LabSOA – Laboratório de Arquitetura Orientado a Serviço

LILACS – Literatura Latino-Americana e do Caribe em Ciências da Saúde

LIS – Laboratório de Informática em Saúde

MEDLINE – *Medical Literature Analysis and Retrieval System Online*

MR – Modelo de Referência

MS – Ministério da Saúde

NCBI – *National Center for Biotechnology Information*

NLM – *National Library of Medicine*

OMS – Organização Mundial da Saúde

OO – Orientação a Objeto

OPAS – Organização Pan-Americana da Saúde

OWL – *Web Ontology Language*

PEP – Prontuário Eletrônico do Paciente

PubMED – Publicações Médicas

RES – Registro Eletrônico em Saúde

SBEB –Sociedade Brasileira de Engenharia Biomédica

SBIS – Sociedade Brasileira de Informática em Saúde

SCIELO – *Scientific Electronic Library Online*

SDK – *Software Development Kit*

SDLC – *Systems Development Life Cycle*

SGBD – Sistema Gerenciador de Banco de Dados

SIS – Sistema de Informação em Saúde

SOA – *Service-Oriented Architecture*

SQL – *Structured Query Language*

SUS – Sistema Único de Saúde

TI – Tecnologia da Informação

TIC – Tecnologia de Informação e Comunicação

TS – *Technical Specification*

UML – *Unified Modeling Language*

UnB – Universidade de Brasília

XML – *eXtensible Markup Language*

XUL – *XML User Interface Language*

1 INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO E FORMULAÇÃO DO PROBLEMA

Existem vários desafios em busca de melhor representar o conhecimento na saúde, visto que a integração dos serviços de saúde e a continuidade do cuidado da população requerem um aprimoramento no uso de Tecnologia de Informação e Comunicação (TIC). Segundo a Sociedade Brasileira de informática em Saúde (SBIS) e o Conselho Federal de Medicina (CFM), o acompanhamento do crescimento da utilização das TICs, demonstram as grandes possibilidades, e benefícios que a informática pode trazer para a área da saúde, dentre elas a interoperabilidade entre os Sistemas de Informação em Saúde (SIS), ou seja, permitir que os SIS sejam mais distribuídos, interconectados, com um alto grau de confiabilidade e segurança a um custo cada vez mais acessível a grandes e pequenas empresas do setor saúde (KONDO, 2012).

Para alcançar sistemas semanticamente interoperáveis e à prova de futuro, a Fundação openEHR criou uma arquitetura à prova de futuro a partir de modelos clínicos livres para uso em qualquer sistema, ou seja, é uma especificação de um modelo de referência com informações pertinentes para a implementação de sistemas de Registros Eletrônico em Saúde (RES) que permite abranger todas as especificidades e peculiaridades dos modelos de domínio em saúde, permitindo o desacoplamento das questões clínicas e tecnológicas, separando a semântica da informação e conhecimento (GAETE, 2012).

A Fundação openEHR é uma empresa sem fins lucrativos, que tem como objetivo principal facilitar a criação e o compartilhamento de registro de saúde de pacientes para a comunidade médica (OPENEHR, 2014). De acordo com Kalra (2006), a Fundação openEHR objetiva também:

- Promover e publicar a especificação de requisitos para representação e comunicação de RES;
- Promover e publicar informação sobre arquiteturas de RES, modelos e dicionários de dados que atendam a esses requisitos;

- Gerenciar a validação das arquiteturas RES por meio da implementação e avaliação clínica;

- Manter implementações de código aberto para melhorar o conjunto de ferramentas disponíveis ao apoio de sistemas clínicos;

- Colaborar com outros grupos de trabalho para a alta qualidade, sistemas de informação de saúde interoperáveis.

De acordo com Santos (2011) o RES pode ser entendido como um repositório de dados do histórico integrado de saúde de um paciente em uma forma processável eletronicamente, armazenado e transmitido com segurança, e acessível por múltiplos usuários ou várias instituições de saúde.

Na busca desta interoperabilidade, a abordagem openEHR, promove a separação do domínio do conhecimento do domínio da informação e é apresentada como solução para o RES. Esta abordagem é chamada de modelagem de dois níveis ou multinível. Separar em dois níveis informação e conhecimento facilita a criação e manutenção desses Sistemas de Informação (SI). Nesta abordagem, é estabelecido um modelo de informação (referência) usado para representar as propriedades genéricas da informação, e um modelo de conhecimento representado pelo conceito de arquétipos. (SANTOS; BAX; PESSANHA, 2010).

Esse conceito pode ser interpretado de duas formas (OPENEHR, 2014):

- Para um profissional da saúde, trata-se de uma definição atômica de um conceito em saúde que expressa tudo o que você deveria querer registrar sobre uma "coisa" em particular, em qualquer situação.
- Para um técnico em SIS, trata-se de uma expressão computável do conteúdo de um domínio na forma de expressões estruturadas limitadas por restrições de um modelo de referência, também vista como um modelo lógico reutilizável.

Entretando, criar uma implementação que siga as especificações do padrão openEHR torna uma matéria trabalhosa na área de Engenharia de *Software* e em função desses interesses, existe um estudo crescente de métricas na busca de soluções para os diversos desafios e problemas relacionados ao desenvolvimento de *software*. A

Fundação openEHR, visto que ela possui um Modelo de Objetos que utiliza os conceitos de OO ela propõe especificações de modelo de objeto de arquétipo que visa detalhar as estruturas de objetos, as classes e seus relacionamentos, e que auxilie na elucidação e definição semântica de arquétipos e modelos baseado na modelagem multinível openEHR.

O principal ponto a ser tratado nesse estudo é a renderização de arquétipos, pois esta sim deve ser levada em consideração em todas as interações entre usuários e sistemas que lidam com informações clínicas e demográficas de pacientes em um ambiente hospitalar, a fim de se alcançar a interoperabilidade semântica na informação.

Do ponto de vista do tratamento das informações do paciente, leva-se como principal ferramenta o uso dos Prontuários Eletrônicos do Paciente (PEP), no qual é um *software* utilizado para gestão das informações e suporte à decisão, onde se organiza e armazena a informação que antes era contida no prontuário em papel, e agora contribuindo com a elegibilidade, a perda das informações, multiplicidade de pastas, deixando as informações dispostas a pesquisas coletivas, padronizadas, e de fácil acesso (GUBIANI, 2003).

Segundo Guabiani (2003), as informações dos PEP ficam dispostas no RES, que se trata de um repositório eletrônico de informações em torno da saúde das pessoas, possibilitando um panorama de seus históricos clínicos, onde diferentes profissionais de saúde (médicos, dentistas, fisioterapeutas, psicólogos...) tem acesso ao mesmo prontuário único, simultaneamente e em diferentes organizações de saúde.

Uma das narrativas clínicas importantes que compõe o PEP é o Sumário de Alta Hospitalar, documento essencial para a continuidade da assistência ao paciente. Este é constituído de informações relacionadas ao acompanhamento do paciente, como evolução clínica, procedimentos assistenciais, intervenções clínicas e diagnósticas, condutas adotadas e iniciadas para seguimento em clínica ou outro estabelecimento de assistência à saúde, e, principalmente, no final de sua permanência hospitalar (LAPELLE et al, 2006).

O objetivo de toda essa especificação é permitir que sistemas de RES alcançassem a interoperabilidade semântica, ou seja, que as informações compartilhadas entre os

mesmos sejam entendidas no nível dos conceitos do domínio formalmente definidos (BITENCOURT JUNIOR, 2009).

Do ponto de vista técnico desse trabalho, Renderização está aliada a busca pela representação gráfica em tela das informações contidas nos arquétipos, sendo estes definidos do contexto do Sumário de Alta Hospitalar.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

Desenvolver um protótipo de *software* através da renderização de arquétipos definidos do Sumário de Alta Hospitalar e representá-los em tela.

1.2.2 Objetivos específicos

- Definir o arquétipo utilizado no estudo de caso;
- Validar o arquétipo definido do Sumário de Alta Hospitalar;
- Montar um algoritimo de uso do *parser* do arquétipo em formato *Archetype Definition Language* (ADL) versão 1.4 para saída em forma de *Archetype Object Model* (AOM);
- Extrair os objetos do arquétipo definido conforme o Modelo de Referência, ADL 1.4 e Modelo de Restrição do AOM openEHR;
- Criar uma representação gráfica dinâmica a partir dos objetos encontrados no arquétipo definido do Sumário de Alta Hospitalar.

1.3 REVISÃO DA LITERATURA

É identificada a necessidade de se fazer uma revisão sistemática na literatura para verificar se há métodos de renderização da *interface* gráfica dos arquétipos. A partir desta necessidade é definido o protocolo para revisão.

A pesquisa da base bibliográfica utilizada neste trabalho considerou a busca por livros, teses, dissertações, monografias, relatórios, revistas da área de Informática em Saúde (IS), Tecnologia da Informação (TI) e artigos, nas seguintes fontes especializadas: Publicações Médicas (PubMed), que é uma base de dados a qual permite a pesquisa bibliográfica de artigos publicados em revistas de grande circulação da área médica, e que foi desenvolvida pelo *National Center for Biotechnology Information*

(NCBI), sendo mantido pela *National Library of Medicine* (NLM); Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES); *Association for Computing Machinery* (ACM); *Institute of Electrical and Electronics Engineers* (IEEE); SBIS; Sociedade Brasileira de Engenharia Biomédica (SBEB); *Science Direct*; Ministério da Saúde (MS) através da Biblioteca Virtual em Saúde (BVS), que é consolidada com a Organização Pan-Americana da Saúde / Organização Mundial da Saúde (OPAS/OMS), e cooperada tecnicamente com a Biblioteca Regional de Medicina (BIREME) em informações científica em saúde na América Latina e Caribe, e integrada por sistemas nacionais que operam redes de bibliotecas e centros de documentação em ciências da saúde.

A BVS disponibiliza consulta nas seguintes bases de dados para literatura técnica científica: Literatura Latino-Americana e do Caribe em Ciências da Saúde (LILACS); *Scientific Electronic Library Online* (SciELO) do Brasil e *Medical Literature Analysis and Retrieval System Online* (MEDLINE), no qual todas são coordenadas pela BIREME. Todas essas bases de dados permitem uma vasta pesquisa bibliográfica, além de disponibilizarem a conexão de pesquisas em outras bases, por exemplo: Repositório Científico do Instituto Politécnico do Porto; Repositório Digital da Universidade de Coimbra; Repositório da Universidade Politécnica de Valência, entre outros.

A pesquisa realizada em todas as bases mencionadas, sem intervalo de anos especificado, com o argumento "*interfaces graphical from openEHR archetypes*" retornou ao total 49 artigos, sendo destes 10 trabalhos tiveram mais relevância, sendo apenas 7 direcionados para a pesquisa proposta. As bases pesquisadas foram nacionais e internacionais.

O artigo "Estudo dos Conceitos presentes em um Resumo de Alta Hospitalar para a Representação das Informações por meio de Arquétipos OpenEHR" (DIAS et al, 2014), descreve o meio de se encontrar a interoperabilidade entre os sistemas por meio de arquétipos openEHR, buscando conceito clínico e demográfico no resumo de internação hospitalar.

O artigo "Registro de Saúde Eletrônico Dinâmico Baseado em OpenEHR" (CESAR et al, 2014), apresenta uma discussão sobre a construção de um sistema de informação em saúde baseado na especificação de arquétipos, proposta pelo modelo de referência

openEHR, cujo objetivo foi de apresentar uma solução em *software* para realizar a leitura de *templates* operacionais e construir dinamicamente a *interface* de acesso do usuário baseado nos *templates*.

O artigo "*Analysis of Metrics for the Usability Evaluation of Electronic Health Record* Systems" (KOPANITSA, TSVETKOVA e VESELI, 2012), trata-se de um método para avaliar a eficácia e a usabilidade dos SIS para que se tenha uma melhor GUI (*Graphical User Interface*), levando em consideração a necessidade de grupos de usuários médicos.

O artigo "*Towards automatically generating graphical user interfaces from openEHR archetypes*" (SCHULER et al, 2006), utiliza os arquétipos openEHR para gerar automaticamente a *interface* para o usuário, através do XUL (idioma de *interface* do Mozila) baseado em XML (*EXtensible Markup Language*), e utiliza o GUI para validação dos dados.

O trabalho de conclusão de curso "*Editor gráfico de correspondências de alto nível para arquétipos de História Clínica Eletrônica*" (RUIZ, 2010), trata da importância da interoperabilidade semântica no RES e o desenvolvimento de uma *interface* visual para fácil especificação declarativa entre os esquemas das fontes de dados e as definições formais existentes de conceitos clínicos padrões para a comunicação baseada em RES.

Foi feito uma revisão sistemática no trabalho de conclusão de curso do Instituto Tecnológico Aeronáutico (ITA), no qual foi criado um editor de *templates* para um RES semanticamente interoperável. Nesse trabalho foi desenvolvido um aplicativo para editar *templates* com especificações openEHR (BITTENCOURT JÚNIOR, 2009).

Também foi escolhida para uma revisão sistemática a leitura do documento técnico, oriundo do MS do Departamento de Informática do Sistema Único de Saúde (SUS), elaborado por (SANTOS 2013). O autor fez a descrição da Linguagem ADL da fundação openEHR em uso na comunidade internacional, com suas principais assertivas e exemplos de utilização (BEALE, 2008).

Os trabalhos analisados propuseram o uso dos arquétipos openEHR na busca principalmente por interoperabilidade entre os sistemas, ou seja, uniformizar a semântica dos conceitos em saúde para facilitar o entendimento da informação.

Também foi proposta a análise e a contextualização de uma *interface* para que o usuário visualizasse as informações dos conceitos clínicos.

A ideia proposta nesse trabalho é utilizar os arquétipos como artefato principal para se atingir interoperabilidade semântica dos conceitos em saúde, no contexto do sumário de alta hospitalar, fazendo com que a informação seja representada em forma de *interface* gráfica para os usuários envolvidos no SIS.

1.4 ORGANIZAÇÃO DO TRABALHO

O trabalho foi dividido em seis capítulos, os quais podem ser resumidos da seguinte forma:

- O primeiro capítulo faz a Introdução e a Contextualização do problema;
- O segundo capítulo apresenta a Fundamentação Teórica do RES, do openEHR, a modelagem dos arquétipos, o modelo de objetos dos arquétipos, do Sumário de Alta Hospitalar, do levantamento teórico para o desenvolvimento de *software* e das ferramentas utilizadas na modelagem;
- O terceiro capítulo apresenta a Metodologia utilizada na pesquisa;
- O quarto capítulo descreve o Estudo de Caso e o resultado obtidos com o desenvolvimento do protótipo;
- O quinto capítulo descreve a Discussão e a Conclusão do trabalho proposto;
- O sexto capítulo apresenta a proposta de Trabalhos Futuros.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 REGISTRO ELETRÔNICO EM SAÚDE

2.1.1 Definição do RES

O RES é composto pelo registro de informação de saúde de um paciente. Porém, esse registro é elaborado a partir de eventos ocorridos em múltiplas organizações de saúde (SANTOS, 2012).

Nas últimas décadas, os sistemas de RES evoluíram muito. O objetivo principal era capturar a maior quantidade de dados do paciente que estavam registrados no papel. Este processo contribuiu com a organização dos dados e das informações de maneira a dar suporte a funções administrativas e de gestão (RUIZ, 2010). O avanço da tecnologia substituiu o que estava no papel para o meio eletrônico. Essa evolução promissora tem evitado a redundância e facilitada às tarefas de tratamento da informação (RUIZ, 2010). Todas as informações do paciente, sendo estas administrativas, financeiras e de dados clínicos, são reunidas dentro de um RES (KONDO, 2012).

De acordo com a fundação openEHR, um RES tem a seguinte definição: "É um registro de cuidado longitudinal, centrado no paciente e compartilhado entre as organizações de atenção à saúde" (BEALE, 2013).

A SBIS e o CFM, através do manual para certificação de sistemas de RES definem, de forma prática, como sendo: "Um repositório de informação a respeito da saúde de indivíduos, numa forma processável eletronicamente" (SILVA, 2011).

A ISO define o RES como:

> [...] um ou mais repositórios, física ou virtualmente integrados, de informação em forma processável por computadores, relevantes para o bem-estar, saúde e atendimento em saúde de um indivíduo, capaz de ser armazenado e transmitido de forma segura e de ser acessível por múltiplos usuários autorizados, representado de acordo com um modelo de informação lógico padronizado e comumente acordado. Seu objetivo principal é o apoio à atenção em saúde ao longo da vida do paciente, de forma integrada, efetiva e segura (ISO/TS 18308, 2004, p. V, tradução nossa).

O PEP é diferente de RES e os dois devem andar em conjunto para o sucesso de objetivos locais, regionais e nacionais para melhorar a seguridade dos dados de um paciente, reduzindo os custos de utilização de saúde (GUBIANI, 2003).

Segundo Gaete (2012), o PEP é o dado do paciente registrado em Hospitais e ambientes ambulatoriais. Este dado é fonte de informação para o RES. Gaete (2012) ainda define que a capacidade de compartilhar informações médicas de maneira fácil entre partes interessadas e de ter informações do paciente é representado pelo PEP.

2.2 A FUNDAÇÃO OPENEHR

2.2.1 Sua Origem

Em 1988, a União Européia estabeleceu a iniciativa *Advanced Informatics in Medicine* (AIM), como parte de seu programa para desenvolvimento de pesquisa e tecnologia. O projeto de pesquisa *Good Electronic Health Record* (GEHR) teve inicio em 1992, como parte da iniciativa AIM, e mais tarde continuou com a participação de grupos de pesquisa da Austrália. Seu objetivo foi analisar abordagens de modelagem de sistemas de RES (OPENEHR *apud* SANTOS e BAX, 2010).

Em 1998, dois dos participantes do projeto GEHR original, Sam Heard e Thomas Beale trabalharam no desenvolvimento de um RES na Austrália, dando continuidade a esse esforço de pesquisa (SANTOS, 2011). Desenvolveu-se assim a ideia de uma arquitetura fundamentada na elaboração de modelagem de dois níveis diferentes: um modelo de informação e um modelo de conhecimento, e dando origem a fundação openEHR.

2.3 MODELAGEM MULTINÍVEL E ARQUÉTIPOS OPENEHR

2.3.1 Modelagem Multinível

De acordo com Beale (*apud* SANTOS, 2011), essa modelagem proporciona a separação da semântica da informação e conhecimento em dois níveis de modelo. No primeiro nível, localiza-se um modelo de informação, também chamado de Modelo de Referência (MR), elaborado a partir da identificação de um pequeno conjunto de classes genéricas e tipos primitivos, suficientes para representar os conceitos que definem e organizam as informações relativas ao RES, isto reflete em características estáveis que

descrevem informações textuais e dados demográficos do paciente, evitando atributos com significados explícitos, deixando as particularidades de conceitos específicos serem representadas no modelo de conhecimento.

Conforme Beale (*apud* GAETE, 2012), o padrão openEHR propõe um MR genérico que permite representar toda a informação gerada no âmbito da atenção à saúde. O núcleo clínico do MR é a classe de entrada (ENTRY). Observando as classes do padrão openEHR da Figura1, observam-se duas classes: CARE ENTRY que corresponde às informações do Cuidado à Saúde e ADMIN ENTRY que corresponde às informações Administrativas. A Figura 1, representa a classe openEHR.

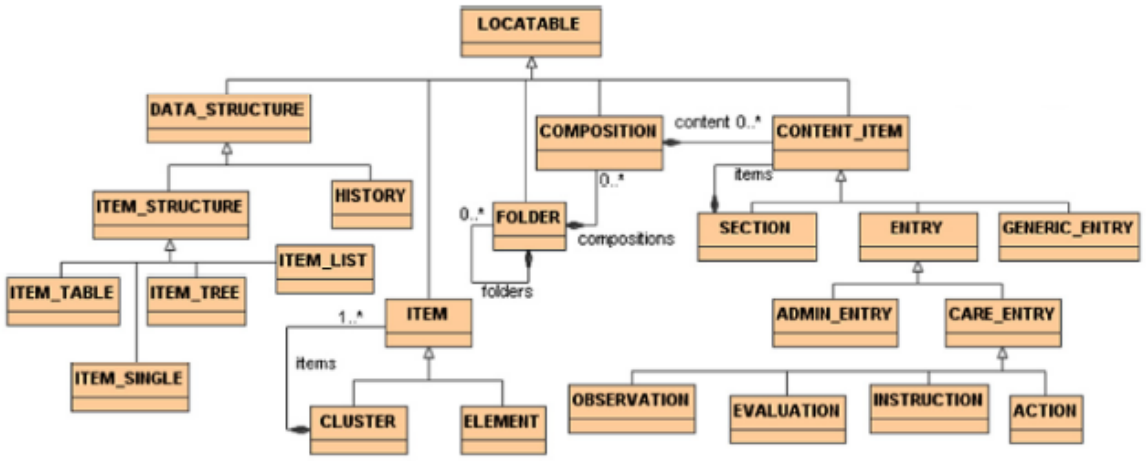


Figura 1: Classes do padrão openEHR (GAETE, 2012).

As descrições das principais classes são vistas como:

- *Observation*: Informação "crua", livre de interpretação de tudo que é dito pelo paciente (sintoma, evento preocupação), achados de exame, resultados de testes ou medidas, e tem como característica principal os registros dos dados mensuráveis, ou observados: pressão arterial, peso, e também ao longo do tempo;

- *Evaluation*: Registra achados interpretados clinicamente, opiniões, resumos, ideias, rotulações ou visões a partir da mente do clínico, interpretados das observações, avaliações, diagnósticos, reações adversas, prognóstico e planos de cuidado;

- *Instruction*: Registros no inicio do processo de *workflow*: prescrição médica, solicitação de exames, também afirmações que podem ser executadas, como receitas de medicamentos, orientações, exames, encaminhamento, entre outros;

- *Action*: Registro da atividade clínica, informação que se registram como resultado da execução de instruções. Ex: Medicação, procedimentos, *etc*.

Essas são as informações de cuidado e dizem respeito ao registro clínico de fato. Seguem outras:

- *Cluster*: Fragmento de arquétipo que pode ser reutilizado em diversos contextos dentro de outros arquétipos. Ex: Sintoma, ciclo menstrual, palpação, *etc.*;

- *Composition*: Corresponde aos documentos clínicos comumente utilizados, sua principal característica é acolher os demais arquétipos eventuais e persistentes. Ex: Resumo de alta, consulta clínica, resumo clínico e relatório cirúrgico;

- *Section*: Elementos que auxiliam a organização e navegação dentro dos RES, e podem estar dentro de outras *sections* (*subsections*). Ex: Exame físico, Sinais vitais, etc.

A modelagem de dois níveis da abordagem é bastante semelhante aos princípios propostos pela área de engenharia de domínio em que há um interesse crescente no campo da engenharia de *software*. Pela engenharia do domínio, destina-se a obter uma descrição clara de um domínio particular, com o objetivo de definir estruturas de dados reutilizáveis para facilitar o desenvolvimento de novos sistemas de informação. Os conceitos de MR do negócio são exemplos claros destas estruturas reutilizáveis (RUIZ, 2010).

Tomando o exemplo discutido no artigo de Thomas Beale "*Archetypes: Constraint-based Domain Models for Future-proof Information Systems"* (BEALE, 2002), no sistema demográfico de um determinado hospital, encontram-se as seguintes entidades em um modelo de referência: pessoas, instalações hospitalares, profissionais de saúde, agente de saúde, funcionário, administrador, paciente, *etc*. No campo dos registros médicos eletrônicos de um modelo de referência deve conter, basicamente:

- Um conjunto de conceitos de negócio, cada um dos quais representa um tipo de bloco de construção da história clínica com granularidade diferente;

- Os tipos de relações possíveis entre os conceitos de negócio (herança, agregação, composição, *etc*.);

- As estruturas de dados, tais como os valores individuais, listas, tabelas, árvores, séries temporais, *etc.*;

- Um conjunto de tipos de dados básicos, como texto, termo codificado, multimídia, quantidade física, gama, *etc*.

O primeiro nível é o utilizado pelos desenvolvedores de TI para modelagem de objetos e esquemas de dados (*data base schemas*) e para manter a informação sempre compreensível é necessário que os conceitos variem muito pouco com o tempo e essa prática facilita também a manutenção do sistema que é implementado em *software*.

No segundo nível, é definido o modelo de conhecimento que especifica de forma semelhante a uma ontologia, e ao mesmo tempo restringe a informação, especificando estruturas de dados clínicos, respectivos tipos de dados e terminologias, efetivamente restringem em uma hierarquia especial de registro as regras de negócios que não são especificadas no modelo de referência. Essas estruturas de dados definidas por um modelo de conhecimento são chamadas de arquétipos. Assim, a fundação openEHR tem influenciado organizações de desenvolvimento de padrões de sistemas de RES.

De acordo com Garde (*apud* SANTOS, 2011), os arquétipos habilitam os especialistas de saúde a definirem formalmente o conteúdo clínico (por exemplo, pacientes, medição de glicose, sumário de alta, etc.); possibilitam maiores chances de conservação do significado dos dados, por meio da especificação bem estruturada e explícita do conteúdo clínico; simplificam o uso de terminologias clínicas; são descritos como um modelo formal e reutilizável de um conceito do domínio. Assim, se um conceito é representado por um arquétipo, este conceito pode ser reutilizado nos vários cenários em que se aplica.

Qamar (2008) afirma que, do ponto de vista de dados, os arquétipos são um meio para fornecimento de semântica para instâncias de dados que estejam em conformidade

com o modelo de referência, assegurando que os dados obedecem a uma estrutura particular e satisfaça a um conjunto de restrições. A descrição semântica dos conceitos de domínio é gerada ligando-se as estruturas de dados e conteúdo a recursos de conhecimento, como terminologias e ontologias.

Na Figura 2, o primeiro nível trata os conceitos específicos do domínio através de terminologias e ontologias, e o segundo, as classes de objetos e os esquemas de dados (BEALE, 2002).

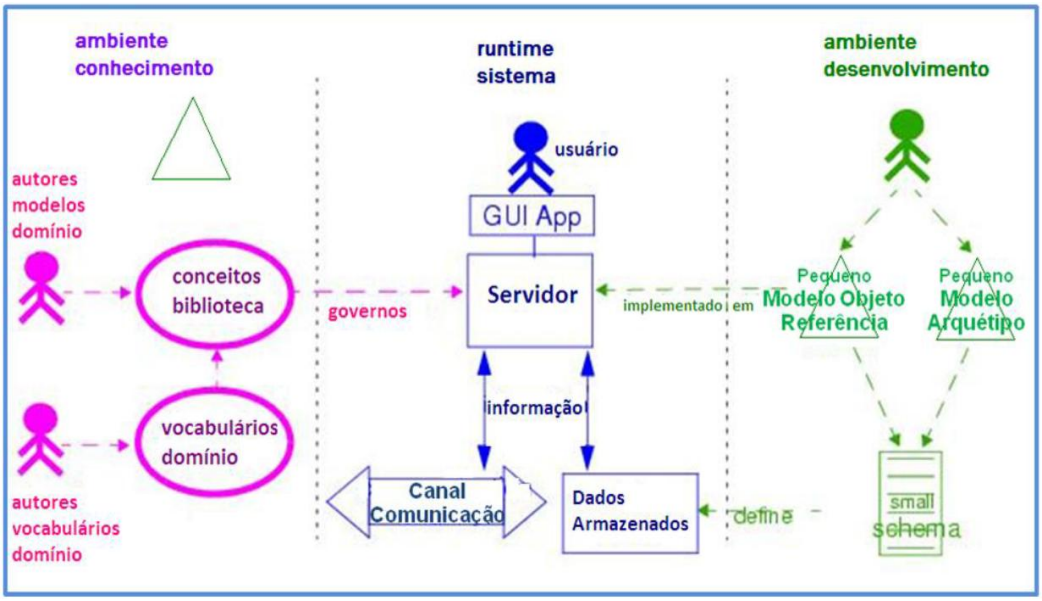


Figura 2: Modelo Dual (adaptado de BEALE, 2002).

Uma das consequências mais interessante do modelo dual é a separação entre a aplicação do sistema de computador e a definição dos conceitos do domínio. Mais especificamente, o *software* ainda é uma implementação do MR e os dados gerenciados pelo sistema são as instâncias do MR, mas os conceitos podem ser definidos por especialistas de domínio, quando o sistema já está em vigor. Consequentemente, sistemas desenvolvidos têm o potencial de ser aplicado mesmo antes de criar os conceitos de domínio, ou seja, a criação de desenvolvimento de *software* independente e, portanto, podem ser específicos para cada sistema ou organização.

A generalidade do MR do primeiro nível é complementada pela especificidade dos arquétipos. De acordo com toda descrição dos arquétipos, observa-se que esse tem que ser criado por especialistas do domínio (médico, enfermeiro, farmacêutico, *etc.*) para formar as definições formais do conteúdo clínico. Esse segundo nível simplifica em

partes o trabalho, por não precisar do conhecimento de especialistas em modelagem de dados em TI, e da expressividade de modelagem de entidades negócios a partir dos arquétipos (BEALE, 2008).

2.3.2 Origem e definição dos Arquétipos

Os arquétipos têm sua origem na antiguidade. Na filosofia pitagórica são modelos matemáticos interpretados como números. Os neoplatônicos utilizaram este termo para definir modelos de toda matéria que existe (KONDO, 2012).

Segundo a normatização da ISO 13606:1, arquétipos são meta-dados usados para definir padrões de modelos clínicos de saúde, incluindo as requisições ou atribuições das diferentes especialidades, serviços e profissões médicas (ISO 13606-1, 2008). Santos (2011) define que dados clínicos estruturados em um modelo de informação restringem um modelo de conhecimento, e estes dados clínicos estruturados são arquétipos.

2.3.3 Especificações dos Arquétipos

O arquétipo é um modelo formal que define um conceito de domínio através de restrições sobre as estruturas de dados contidos no MR subjacente.

As restrições podem ser vistas como especificações que determinam as características que devem atender ao pedido de um MR, ou seja, um conceito de negócio válido para um determinado domínio. Os tipos básicos de restrições (GAETE, 2012):

- Restrições ao domínio atributos como uma declaração do máximo e/ou valor mínimo ou uma lista de valores aceitos;

- Restrições sobre as relações de agregação entre arquétipos. Isso inclui a especificação das condições a serem cumpridas para um arquétipo que podem estar contidos em outros, como a cardinalidade da relação, ou seja, quantas instâncias do conteúdo arquétipo podem aparecer dentro de um arquétipo de instância recipiente;

- Restrições sobre atributos obrigatórios e o número de ocorrências possíveis se forem vários valores.

Assim, através dos arquétipos é proporcionada uma capacidade muito rica e completa de especificações de compartilhamento das informações clínicas.

2.3.4 Archetype Definition Language (ADL)

A ADL é uma linguagem formal para expressar os arquétipos e com sintaxe de serialização dos arquétipos. A linguagem se baseia em modelos de restrições de entidades de domínio, ou "regras de negócio estruturadas". Utiliza-se de três diferentes sintaxes: *Data* ADL (dADL) ou modo de definição de dados do ADL, *Constraint* ADL (cADL) ou modo de restrições, e uma versão de *First-Order Predicate Logic* (FOPL) ou versão de lógica de predicados de primeira ordem, para descrever restrições em dados que são instâncias de algum modelo de informação) (BITTENCOURT JÚNIOR, 2009).

2.3.5 Estrutura ADL

Arquétipos expressos em ADL lembram arquivos de linguagens de programação, e possui uma sintaxe própria para agrupar elementos (GAETE, 2012). A estrutura em alto nível de um arquétipo em ADL é mostrada na Figura 3 (BITTENCOURT JÚNIOR, 2009).

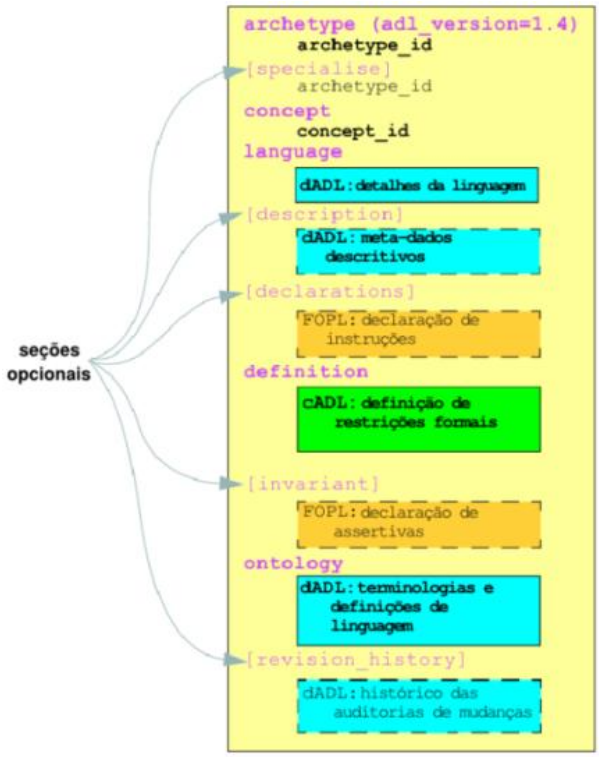


Figura 3: Estrutura da linguagem ADL 1.4 (SANTOS, 2013).

Resumidamente, um código ADL apresenta as seguintes seções básicas na estrutura do código (SANTOS, 2013):

- *Header* – O cabeçalho inicial do arquétipo é descrito nesta seção acrescentando a identificação do arquétipo e do conceito clínico em evidência. Especialização (Herança de arquétipos), linguagem principal, traduções, informações da autoria, descrevem o arquétipo e Subseções: *archeytpe, specialise, concept, language, translation e description;*

- *Definition* – As restrições são representadas em cADL e nesta seção são definidas as restrições para os objetos e atributos presentes no arquétipo;

- *Invariant* – Escrita em FPOL, esta seção tem uma versão de lógica de predicados de primeira ordem para restrições e que contem fórmulas matemáticas ou lógicas. Pode-se utilizar através de *paths* o acesso aos objetos e estruturas;

- *Ontology* – Codificada em dADL esta seção determina a configuração e nome dos objetos, além de links e restrições e Subseções;

- *Revision_history* – Esta seção codificada em dADL é opcional e define o histórico de alterações ocorridas no arquétipo.

2.3.6 dADL (*Data* ADL)

De acordo com Santos et al (2010), o dADL serve para expressar instâncias de dados baseados em um modelo de informação, e sua estrutura é legível para humanos e computadores. As seções do código ADL: *language e translation, description, terminologies_available, term_definitions, constraint_definitions, term_binding e constraint_binding* utilizam esta sintaxe. Um documento dADL pode trabalhar com um ou mais objetos de um mesmo modelo de objetos e todos os seus identificadores são oriundos de um MR (informação). Dois tipos de identificadores são utilizados: nomes de tipos e nomes de atributos.

dADL é uma sintaxe muito simples para expressar dados, se essas instâncias de um MR (resumo da história no nosso caso) ou meta- informação contida na definição de

arquétipos. O modelo de dados subjacente é puramente estruturas de dados hierárquicos são construídas a partir do modelo geral: Identificador_atributo = <valor>.

Em dADL não é necessário indicar o tipo de um atributo quando é primitivo , apenas especificar o valor , assim, uma maior independência é alcançado com o respeito aos nomes e estrutura dos tipos primitivos utilizados no MR. Um atributo identificador pode ser acompanhado por um qualificador, que deveria ser exemplo comparável de qualquer tipo básico, por exemplo: o endereço ("home") ou o endereço ("trabalho"). Atributos qualificados podem representar listas ou tabelas de valores, independentemente de como essas estruturas são definidas na origem e o destino dos SI. dADL também permite um identificador para associar as ocorrências de um tipo com a estrutura:

TIPO_A [id_1] = <Identificador_atributo = <valor1>

Onde construir uma instância do tipo "TIPO_A" e associar um identificador "Id_1", fará referência neste caso em outros lugares e, assim, reutilizar informações (RUIZ, 2010).

Identificadores de termos começam com uma letra minúscula, enquanto nomes de objetos e sua descrição começam com uma letra maiúscula. Um exemplo é mostrado na Figura 4.

```
["pt-br"] = <
        items = <
            ["at0000"] = <
                text = <"Objeto 1"> -- Mostrando a definição do termo objeto 1
                description = <"Descrevendo o objeto 1">
            >
            ["at0001"] = <
                text = <"Objeto 2"> -- Mostrando a definição do termo objeto 2
                description = <"Descrevendo o objeto 2">
            >
```

Figura 4: Exemplo de trecho em código dADL (SANTOS, 2013).

2.3.7 cADL (*Constraint* ADL)

O cADL é uma sintaxe que aplica restrições sobre os dados definidos por modelos de informação orientados a objetos. Pode ser utilizado em tempo de desenvolvimento, por autores e ferramentas, e em tempo de execução por algoritmos computacionais para validação de instâncias de dados. Dentre suas palavras reservadas destacam-se: *matches*

que define uma relação de pertinência; *ocurrences* que indica o número de instâncias possíveis de um atributo; *existence* que indica a opcionalidade de um atributo; *cardinality* que define se o atributo é um container e as *palavras ordered, unorderede e unique* que definem o tipo de agrupamento lógico usado no container. Expressões regulares podem ser incorporadas para validação de *strings* (SANTOS e BAX, 2010).

Assim como acontece com dADL, identificadores do modelo de informação subjacente são usados para todos os *nodes* em cADL, e obedecem as mesmas regras em dADL. Nomes de tipo e de qualquer propriedade (atributo ou função) são permitidos (BEALE e HEARD, 2014). É preciso apenas declarar restrições para as partes do MR que precisa ser restrito, portanto, pode ser mais identificadores de atributos, tipos e relacionamentos do MR utilizado nos arquétipos. Veja Figura 5.

```
EVALUATION[at0000] matches {-- Sintomas Clínicos
        data matches {
                ITEM_TREE[at0001] matches {    -- Lista
                        items cardinality matches {1..*; ordered} matches {
                                ELEMENT[at0002] matches {      -- Resumo
                                        value matches {
                                                DV_TEXT matches {*}
                                        }
                                }
                        }
                }
        }
}
```

Figura 5: Exemplo de código em cADL (SANTOS, 2013).

As restrições que definem um arquétipo não podem ser mais forte do que os do MR, assim, por exemplo, um atributo obrigatório no MR não pode ser opcional em um arquétipo (RUIZ, 2010).

As restrições cADL definidas por cláusulas que restrinjam as instâncias dos tipos e valores de atributos, são estruturados em blocos aninhados que começam com um nome de tipo (classe) ou um nome de atributo do MR. A estrutura geral de um cADL é:

entidade { Restrição }

Onde entidade é um nome de tipo ou atributo.

2.3.8 Assertion

Assertion é uma versão de FOPL que é uma linguagem de lógica de predicados de primeira ordem que vem inserida em cláusulas nas seções *Declaration*, instruções declarativas, *Invariant* e instruções assertivas. Em especial, se utiliza de operadores de comparação e igualdade. Faz partes das cláusulas presentes na seção *definition* e seção *invariant*. Combina FPOL, atribuições e operadores de comparação (=, >, etc) (BEALE; HEARD, 2014). Ver Figura 6.

```
soma_formula:
                /data[at0002]/events[at0003]/data[at0001]/items[at0021]/value/magnitude =
                /data[at0002]/events[at0003]/data[at0001]/items[at0003]/value/value +
                /data[at0002]/events[at0003]/data[at0001]/items[at0004]/value/value +
                /data[at0002]/events[at0003]/data[at0001]/items[at0005]/value/value +
```

Figura 6: Exemplo de trecho de código *Assertion* (SANTOS, 2013).

2.3.9 Relação entre o Modelo de Referência e o Modelo de Conhecimento (Arquétipos)

Neste ponto do texto é importante ter clara a relação entre os MR e os Arquétipos, e como esses modelos são instanciados. Na Figura 7 pode-se ver que as informações são instâncias de um MR, e os arquétipos são instâncias de um modelo de conhecimento. Este, por sua vez, é formalmente vinculado ao modelo de informação. As informações são restringidas em tempo de execução pelos arquétipos. Os conceitos do domínio de saúde são representados na forma de arquétipos (SANTOS, 2012).

A Figura 7 ilustra o relacionamento entre o MR, modelo de conhecimento (arquétipos) e os dados (instâncias).

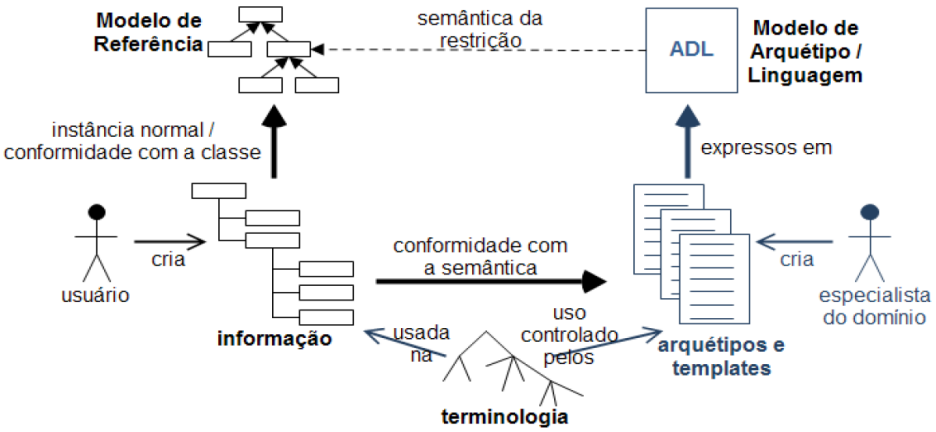


Figura 7: Arquitetura da relação entre a informação e o conhecimento (MENEZES, 2011).

2.3.10 Função Slot

Slot ou ranhura no arquétipo raiz é uma declaração prospectiva dentro do arquétipo raiz, onde outros arquétipos podem (ou devem) ser inclusos neste momento na composição, esse mecanismo também favorece o reuso do conhecimento. São declarações de restrições sobre o que pode ser sensato para adicionar nesse ponto particular. É evidente que o próprio MR é limitado - seções só podem conter outras seções ou entradas, estruturas de árvore só podem conter grupos ou elementos e as listas de elementos únicos. *Slot* é ir além desses limites para dizer o que arquétipos são sensíveis a adicionar a este ponto, mas o arquétipo que utilizar a função *slot* deve ser do tipo apropriado (para um elemento do *cluster* só é permitido para ser adicionado dentro de uma entrada, por exemplo) (OPENEHR, 2014a).

2.3.11 Algoritmo de Resolução de Referências no Arquétipo

Para o uso do arquétipo com a função *Slot*, é preciso utilizá-lo como um arquétipo de entrada, tipo raiz, e resolver todas suas referências, para ter como saída um mapa de arquétipoID.

Para se atingir esse objetivo, foi criado o seguinte algoritmo:

- Encontre todos os *slots* no arquétipo de entrada.
- Para cada *slot*:
    - Olhe-se para os arquétipos correspondentes em cada repositório;
    - Se houver mais do que um arquétipo para o mesmo conceito (por exemplo, mais do que uma versão):
        - Será necessário selecionar o arquétipo de direito ou,
        - A última versão será selecionada por padrão (isto não poderia ser feito se a última versão é um ramo).
    - Para cada arquétipo referenciado:
        - Invocar *recursivelly*, ou seja, sempre invocar uma função de recursividade com o arquétipo;

- Adicione o arquétipo para o mapa de arquétipoID.

2.4 MODELO DE OBJETOS OPENEHR

2.4.1 Modelo de Objetos de Arquétipos

A fundação openEHR fornece especificações de SIS e mecanismos de interoperabilidade utilizando notações que representam conceitos da abordagem de OO. A OO é uma abordagem para desenvolvimento de *software* que trata os problemas do mundo real como um conjunto de objetos distintos. Leva-se em conta a estrutura e o comportamento dos dados para esta representação (ALKIMIM, 2013).

A fundação OpenEHR também dispõem de um AOM. O AOM é um modelo de objeto de arquétipo, tipo genérico que pode ser usado para expressar arquétipos de qualquer MR. Este modelo pode ser usado como base para desenvolvimento de sistemas que se baseiam na modelagem multinível, e que processam arquétipos de forma independente, definindo um modelo de objetos como um resultado de uma modelagem *Unified Modeling Language* (UML) (BEALE, 2008). A UML é um padrão estabelecido para a modelagem de sistemas, sendo específica para modelagem OO, proporcionando um desenvolvimento de maneira iterativa, com visualização das funcionalidades e organização dos objetos no sistema.

Na UML o mundo real do usuário é representado na forma de diagramas, de maneira que as tarefas dos usuários serão apresentadas como funcionalidades do sistema (BORGES e MORO, 2013). É muito importante que o AOM seja compreensível e implementável por pessoas técnicas, tendo em vista que, especificações formais de padrões de informação em saúde, são destinados a usuários técnicos desenvolvedores de sistema de informação. Dessa forma o padrão UML foi adotado como modelo de gráfico (BEALE, 2007).

O AOM representa, através de arquétipos, os conceitos específicos de dados clínicos. Um arquétipo descreve a configuração de instância de dados de classes do MR. O AOM é relacionado com o MR, de maneira que sua semântica são de restrições em instâncias de classes definidas no MR. Se os dados são criados e modificados usando arquétipos, então estes arquétipos restringem a configuração dos dados de acordo com o arquétipo (ALKIMIM, 2013).

Há também nós que representam as referências internas para outros nós, chamados de nós de restrição de referência, o qual referem a uma restrição de texto do atributo *constraint_binding* da seção *ontology* de um arquétipo, e nós de restrição de arquétipo, que representam restrições sobre outros arquétipos que possuem permissão para aparecer em um determinado ponto (SANTOS, 2013).

Beale (2008) detalha os seguintes tipos de nós:

- *C_complex_object*: qualquer nó interior representando uma restrição em instâncias de tipos não primitivos por exemplo: *ENTRY e SECTION*;
- *C_attribute*: um nó representando uma restrição no atributo em um tipo de objeto;
- *C_primitive_object*: um nó representando uma restrição em um tipo de objeto primitivo;
- *Archetype_internal_ref*: um nó que refere para outro nó de objeto definido previamente no mesmo arquétipo;
- *Constraint_ref*: nó que refere-se a uma restrição, normalmente, em um texto ou entidade de termo codificado. Na linguagem ADL é representada com um código do tipo "acNNNN". A restrição é expressa como uma consulta em uma entidade externa, normalmente uma terminologia ou ontologia;
- *Archetype_slot*: um nó cujas declarações definem uma restrição que determina que outros arquétipos podem aparecer naquele ponto no arquétipo atual. Uma analogia pode ser o buraco de uma fechadura, em que poucas ou muitas chaves podem encaixar, dependendo do formato específico. Tem a mesma semântica de um C_COMPLEX_OBJECT, exceto que as restrições são expressas em outro arquétipo.

Beale (2008) exibe o modelo dividido nos seguintes pacotes:

- Pacote *Archetype*: este contém apenas duas classes: *ARCHETYPE e VALIDITY_KIND*. Entretanto, a classe *ARCHETYPE* é a principal classe dentro do AOM. Um arquétipo é modelado como uma especialização de *AUTHORED_RESOURCE* e inclui meta-dado descritivo, informação de idioma e histórico de revisão;

- Pacote *Constraint Model*: com base nas classes deste pacote é possível criar estruturas de descrição do arquétipo que são uma alternância hierárquica de objeto e atributo. Trata-se de uma consequência direta do princípio de OO, onde classes consistem em propriedades, que por sua vez têm tipos que são classes;
- Pacote *Assertion*: contém classes utilizadas para restringir dados dentro do arquétipo. Utiliza expressões lógicas são representadas nos arquétipos em lógica de predicados de primeira ordem (*First-Order predicate Logic - FOPL*);
- Pacote *Primitive:* qualquer definição de arquétipo delegará para o nó folha restrições em instâncias de tipos primitivos. Em outras palavras, os tipos de dados descritos aqui, veja os princípios de OO, definidos em UML, tomando cuidado para não invalidar sua implementação em qualquer linguagem de programação. Harmonização de tipos de dados (*data types*) entre modelos de informação é essencial para reduzir o trabalho de conversão e possíveis erros entre serviços de infraestrutura de informação em saúde, bem como os tipos de dados são projetados para trabalhar com todos os modelos definidos pelo openEHR.
- Pacote *Ontology*: possui classes para representar a ontologia de um arquétipo. Uma ontologia de arquétipo que consiste em lista de termos definidos para o arquétipo, lista de definição de restrição externa e um conjunto de uma ou mais ligações de definições de termos para códigos de terminologias externas.

Todas as classes dos pacotes do AOM estão dispostas em diagrama, de forma que fique representada todas as classes e suas relações, identidade, propriedades (atributos) e métodos (comportamento). As classes podem estar ligadas a outras classes através de associações que indicam cardinalidade, classificação, navegabilidade, generalização como fator de compartilhamento da estrutura e comportamento caracterizado como herança. O diagrama de classes do pacote do Modelo de Restrição é apresentado na Figura 8.

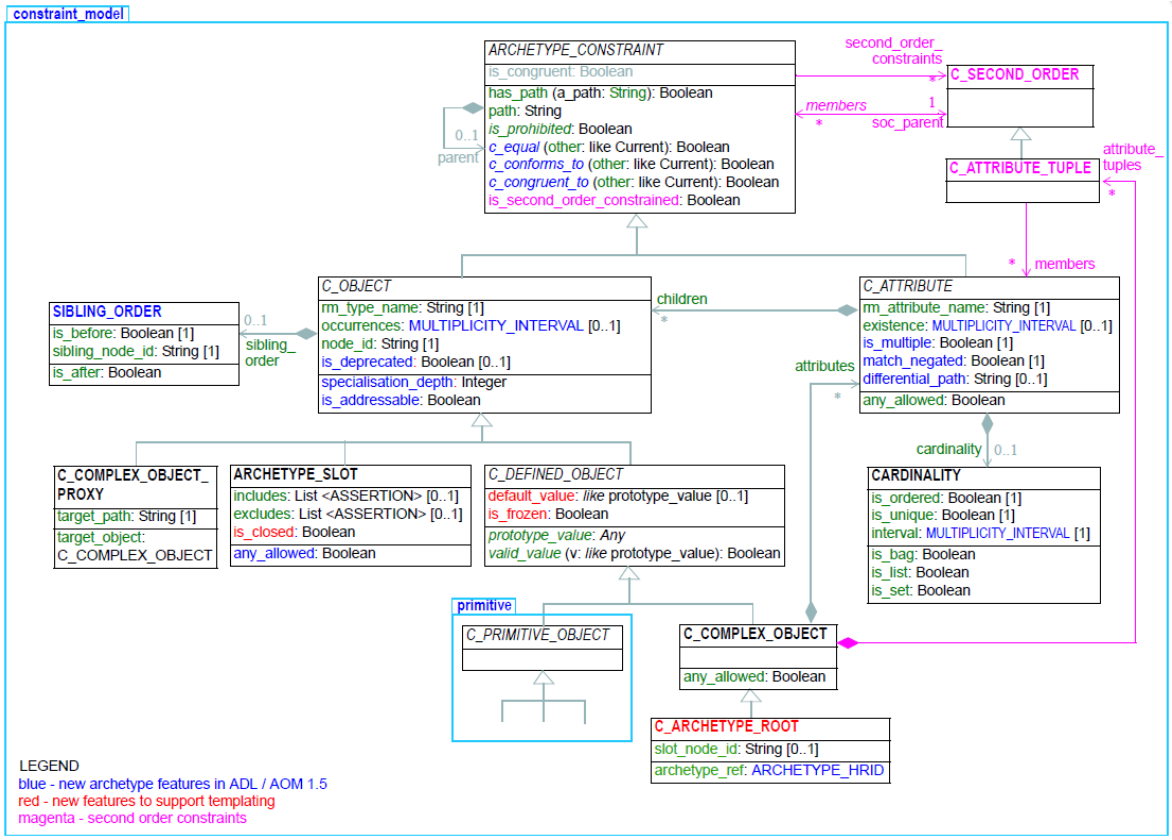


Figura 8: Diagrama de Classes do Modelo de Restrição de Arquétipos openEHR (OPENEHR, 2014).

2.4.2 Parser ADL Java

Em ciência da computação é conhecido como análise sintática (também conhecida pelo termo em inglês *parsing*) é o processo de analisar uma sequência de entrada (lida de um arquivo de computador ou do teclado, por exemplo) e transforma a entrada em uma estrutura de dados, em geral uma árvore, o que é conveniente para processamento posterior e captura a hierarquia implícita desta entrada (OPENEHR, 2014b).

O analisador sintático ou *parser* é um objeto que faz análise sintática de um texto para determinar a sua estrutura lógica, ou seja, transforma uma *string* em um valor de um determinado tipo, conforme a Figura 9.

Figura 9: *Parser*

Baseado nas especificações openEHR, o processo de *parsing* implica em mudar a expressão sintática de um arquétipo (ADL, XML, OWL) em um objeto. O arquivo de entrada é convertido em uma árvore de objetos (SANTOS, 2013).

A Figura 10 demonstra o *parse* ADL Java.

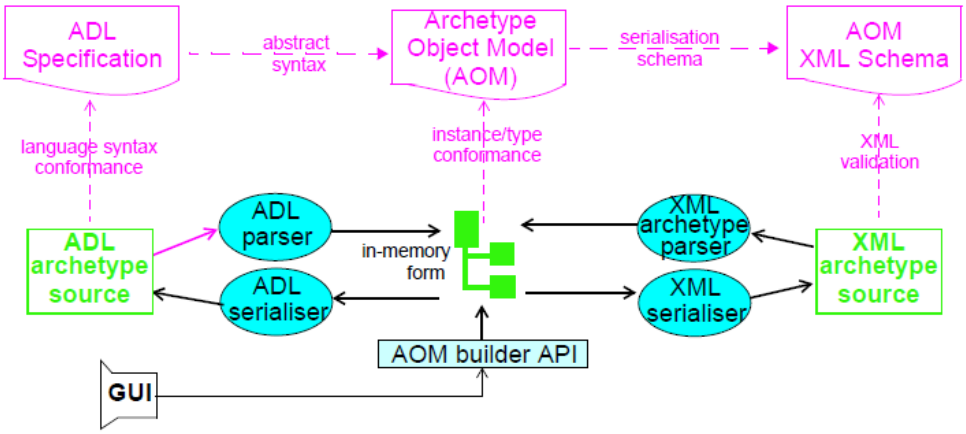


Figura 10: Estrutura do *parse* ADL Java.

Esta é uma implementação Java de um analisador que pode ler arquétipos openEHR no formato ADL e tem como saída a árvore de objetos em forma de AOM.

2.5 REPRESENTAÇÃO DO CONHECIMENTO

2.5.1 Sumário de Alta Hospitalar

Relatório clínico resumido elaborado por um médico ou outro profissional por ocasião do término do período de internação, conclusão de uma série de tratamentos feitos em nível hospitalar, incluindo emergência, hospital dia e internação domiciliar. Ele contém informações sobre motivo da admissão, diagnósticos e outros achados importantes, exames realizados e tratamentos administrados, resposta e eventuais reações do paciente aos tratamentos administrados, intercorrências clínicas, prognóstico e conduta e recomendações escritas para seguimento pós-alta (DIAS, 2014).

O propósito do Sumário de Alta Hospitalar é o de comunicar de forma resumida as informações clínicas mais relevantes de um episódio de cuidado de doenças agudas ou agudizações de doenças pré-existentes em unidade de cuidados secundários: hospital, hospital dia, internação domiciliar ou em setor de emergência, quando de sua conclusão com o objetivo de prover informações acuradas e oportunas para a equipe ou profissional que vai dar seguimento ao tratamento do paciente, possibilitando com isso a continuidade do cuidado do paciente (CEE-IS, 2014).

A Figura 11 ilustra o sumário de alta sendo gerado após uma internação e enviado para ser utilizado na continuidade do cuidado do paciente.

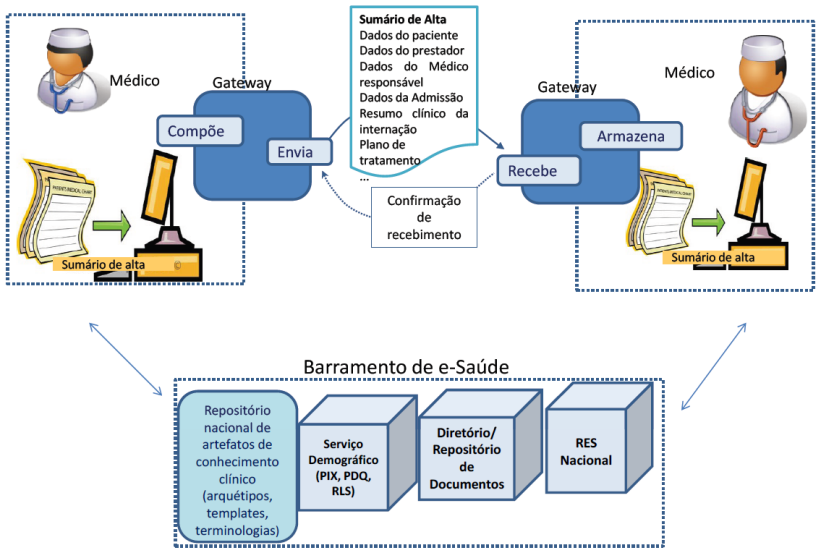
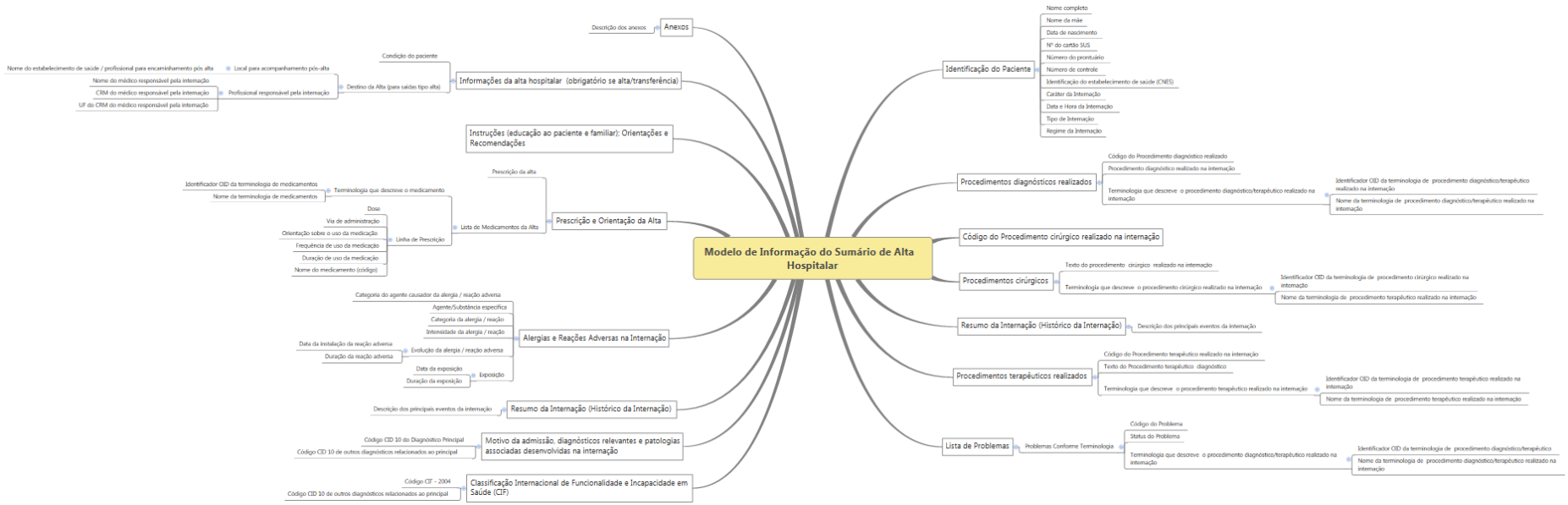


Figura 11: Fluxo do Sumário de Alta Hospitalar (CEE-IS, 2014).

2.5.2 Modelo de Informação do Sumário de Alta Hospitalar

O 1º Projeto de Norma do sumário de alta hospitalar foi elaborado pela Associação Brasileira de Normas Técnicas / Comissão de Estudo Especial de Informática em Saúde (ABNT/CEE-IS 78) em Fevereiro de 2014 e tem como identificação a numeração 78:000.01-021, a mesma será diagramada conforme as regras de editoração da ABNT quando for publicada como Norma Brasileira, portanto ainda não possui valor normativo (CEE-IS, 2014). A Figura 12 ilustra esta abordagem .


Modelo de Informação do Sumário de Alta Hospitalar
Identificação do Paciente
Nome completo
Nome da mãe
Data de nascimento
Nº do cartão SUS
Número do prontuário
Número de controle
Identificação do estabelecimento de saúde (CNES)
Caráter da Internação
Data e Hora da Internação
Tipo de Internação
Regime da Internação
Procedimentos diagnósticos realizados
Código do Procedimento diagnóstico realizado
Procedimento diagnóstico realizado na internação
Terminologia que descreve o procedimento diagnóstico/terapêutico realizado na internação
Identificador OID da terminologia de procedimento diagnóstico/terapêutico realizado na internação
Nome da terminologia de procedimento diagnóstico/terapêutico realizado na internação
Código do Procedimento cirúrgico realizado na internação
Procedimentos cirúrgicos
Texto do procedimento cirúrgico realizado na internação
Terminologia que descreve o procedimento cirúrgico realizado na internação
Identificador OID da terminologia de procedimento cirúrgico realizado na internação
Nome da terminologia de procedimento terapêutico realizado na internação
Resumo da Internação (Histórico da Internação)
Descrição dos principais eventos da internação
Procedimentos terapêuticos realizados
Código do Procedimento terapêutico realizado na internação
Texto do Procedimento terapêutico diagnóstico
Terminologia que descreve o procedimento terapêutico realizado na internação
Identificador OID da terminologia de procedimento terapêutico realizado na internação
Nome da terminologia de procedimento terapêutico realizado na internação
Lista de Problemas
Problemas Conforme Terminologia
Código do Problema
Status do Problema
Terminologia que descreve o procedimento diagnóstico/terapêutico realizado na internação
Identificador OID da terminologia de procedimento diagnóstico/terapêutico
Nome da terminologia de procedimento diagnóstico/terapêutico realizado na internação
Anexos
Descrição dos anexos
Informações da alta hospitalar (obrigatório se alta/transferência)
Condição do paciente
Destino da Alta (para saídas tipo alta)
Local para acompanhamento pós-alta
Nome do estabelecimento de saúde / profissional para encaminhamento pós alta
Profissional responsável pela internação
Nome do médico responsável pela internação
CRM do médico responsável pela internação
UF do CRM do médico responsável pela internação
Instruções (educação ao paciente e familiar): Orientações e Recomendações
Prescrição e Orientação da Alta
Prescrição da alta
Lista de Medicamentos da Alta
Terminologia que descreve o medicamento
Identificador OID da terminologia de medicamentos
Nome da terminologia de medicamentos
Linha de Prescrição
Dose
Via de administração
Orientação sobre o uso da medicação
Frequência de uso da medicação
Duração de uso da medicação
Nome do medicamento (código)
Alergias e Reações Adversas na Internação
Categoria do agente causador da alergia / reação adversa
Agente/Substância específica
Categoria da alergia / reação
Intensidade da alergia / reação
Evolução da alergia / reação adversa
Data da instalação da reação adversa
Duração da reação adversa
Exposição
Data da exposição
Duração da exposição
Resumo da Internação (Histórico da Internação)
Descrição dos principais eventos da internação
Motivo da admissão, diagnósticos relevantes e patologias associadas desenvolvidas na internação
Código CID 10 do Diagnóstico Principal
Código CID 10 de outros diagnósticos relacionados ao principal
Classificação Internacional de Funcionalidade e Incapacidade em Saúde (CIF)
Código CIF - 2004
Código CID 10 de outros diagnósticos relacionados ao principal

Em agosto de 2011, o MS publicou a Portaria 2073 que regulamenta o uso de padrões de interoperabilidade em SIS no âmbito do SUS**.** Esta portaria visa, entre outros objetivos, apoiar a implementação de um RES nacional e longitudinal (BRASIL, 2011). A CEE-IS ciente deste processo ofereceu sua contribuição através da proposta de conteúdo de informação do Sumário de Alta Hospitalar. Esta proposta foi organizada em dois documentos: Modelo de Informação (Anexo 1) e Modelo Computacional.

O Modelo de Informação do sumário de alta hospitalar definiu o conjunto de informações requeridas em um sumário de alta genérico, produzido por ocasião da alta de um paciente de uma internação hospitalar. O destinatário principal desse documento é o médico ou a equipe de referência que acompanhava o paciente antes do episódio de internação e que irá receber o paciente e continuar seus cuidados. Esse destinatário pode ser o médico de família, clínico geral, pediatra, psiquiatra, geriatra ou a equipe de cuidadores do paciente de um centro de saúde da família, de atenção ao idoso ou psicossocial, etc. Esse documento, portanto, contem o relato resumido, porém completo, de todo o período de internação, que permita esse profissional ou equipe prosseguir com a atenção ao paciente e lhe prover o cuidado mais adequado à suas condições clínicas. Para isso ele inclui informações relevantes sobre o paciente, detalhes sobre o tipo e regime de internação, condições de sua admissão e alta, a evolução clínica durante a internação os problemas apresentados, diagnósticos realizados, as medicações que foram utilizadas e modificadas durante a internação, além das intercorrências relevantes, tais como os efeitos adversos e alergias surgidas durante o período de internação. O modelo de informação encerra-se com o plano de tratamento do pós alta, incluindo a lista completa das medicações prescritas com respectiva posologia e outras recomendações (CEE-IS, 2014).

O Modelo Computacional do sumário de alta hospitalar, ainda não foi elaborado, portanto evidenciará as *interfaces* computacionais para que se estabeleça a interoperabilidade entre os SIS no País. Os padrões computacionais a serem adotados serão aqueles já descritos na Portaria 2073 (BRASIL, 2011).

2.6 LEVANTAMENTO TEÓRICO PARA DESENVOLVIMENTO DO PROTÓTIPO DO SISTEMA

2.6.1 Engenharia de Software

A Engenharia de *Software* trata de aspectos relacionados ao estabelecimento de processos, métodos, técnicas, ferramentas e ambientes de suporte ao desenvolvimento de *software*, visando a qualidade dos produtos de *software* e aumentar a produtividade no processo de desenvolvimento (SOMMERVILLE, 2007).

2.6.2 Ciclo de Vida de Desenvolvimento

Para alguns sistemas, o ciclo de vida não é longo, portanto é necessário entender que um *software* nunca estará totalmente acabado; ele poderá estar pronto para uso, mas sempre sofrerá implementações e melhorias (REZENDE, 2005).

O Ciclo de Vida do *Systems Development Life Cycle* (SDLC), conhecido também como o "ciclo de vida do *software*" refere-se aos estágios de concepção, projeto, criação e implementação de um SI. Um desdobramento possível para o SDLC é mostrado na Figura 13.

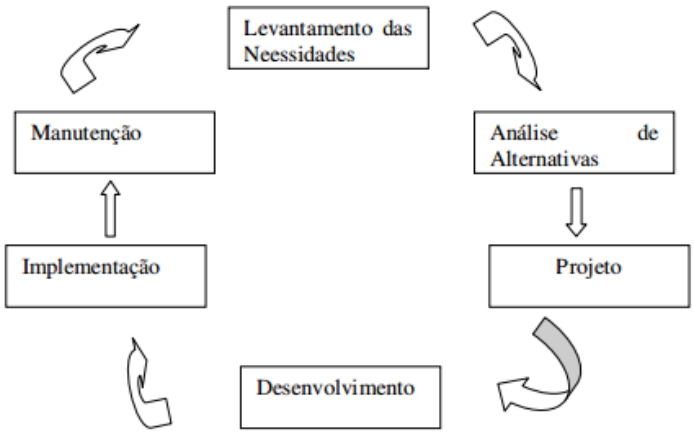


Figura 13: Ciclo de vida do desenvolvimento de sistemas (PRESSMAN, 1995).

A seguir, são descritas cada etapa da Figura 13, isto é:

- O levantamento das necessidades, também chamado de análise de requisitos, identifica as necessidades de informações da organização;

- A análise de alternativas consiste na identificação e avaliação de sistemas alternativos;

- Projeto trata da construção das especificações detalhadas para o projeto selecionado. Essas especificações incluem o projeto das *interfaces*, banco de dados, características físicas do sistema, tais como número, tipos e localizações das estações de trabalho, *hardware* de processamento, o cabeamento e os dispositivos de rede. Deve especificar os procedimentos para testar o sistema completo antes da instalação;

- Desenvolvimento inclui o desenvolvimento ou aquisição do *software*, a provável aquisição do *hardware* e o teste do novo sistema;

- Implementação ocorre após o sistema ter passado satisfatoriamente por testes de aceitação. O sistema é transferido do ambiente de desenvolvimento para o ambiente de produção. O sistema antigo (se existir) deve migrar para o novo;

- Manutenção refere-se a todas as atividades relacionadas a um sistema depois que ele é implementado. Deve incluir atividades tais como a correção de *software* que não funcione corretamente, a adição de novos recursos aos sistemas em resposta às novas demandas dos usuários, entre outros.

Não há modelo de SDLC uniformemente aceito. Alguns modelos combinam desenvolvimento e implementação em uma única etapa. Outros combinam o levantamento e a análise das necessidades também em uma única etapa. Alguns modelos dividem o projeto em lógico e físico (BROOKSHEAR, 2003pre).

O ciclo de vida de um *software* sempre estará em desenvolvimento, pois no decorrer de seu uso haverá sempre a necessidade de modificações (BROOKSHEAR, 2003).

A fase de Manutenção concentra-se nas:

- Correções: Mudanças que estão associadas à correção de defeitos (Manutenção Corretiva);

- Adaptações: são exigidas à medida que o ambiente do sistema evolui e (Manutenção Adaptativa);
- Melhoramento Funcional produzidas por exigências variáveis do cliente (Manutenção perfectiva).

Prevenção:

- O *software* de computador se deteriora devido a modificações, e;
- Surge a manutenção preventiva, frequentemente chamada de reengenharia de *software*;
- Em essência, a manutenção preventiva faz modificações nos programas de computador, de modo que eles possam ser mais facilmente corrigidos, adaptados e melhorados.

2.6.3 Prototipação

Protótipos são representações de um sistema construído antes que os cenários finais venham a existir de fato (BUCHENAU e SURI, 2000). Eles são muito úteis na discussão de ideias com *stakeholders*. São dispositivos que facilitam a comunicação entre os membros da equipe e consistem em uma maneira eficaz de testar as ideias (PREECE, ROGERS e SHARP, 2005). A propotipação permite a criação, a avaliação e o refinamento de opções de *design*, até que se alcance a usabilidade desejada.

Um protótipo é uma versão inicial de um sistema de *software*, que é utilizada para mostrar conceitos, experimentar opções de projeto e, em geral, para conhecer mais sobre os problemas e suas possíveis soluções. O desenvolvimento rápido de um protótipo é essencial para que os custos sejam controlados e os usuários possam fazer experiências com o protótipo no início do processo de *software* (BROOKSHEAR, 2003).

Um protótipo de *software* apoia duas atividades do processo de engenharia de requisitos (SOMMERVILLE, 1992):

- Levantamento de requisitos - Os protótipos de sistema permitem que os usuários realizem experiências para ver como o sistema apóia seu trabalho. Eles obtêm novas ideias para os requisitos e podem identificar pontos positivos e negativos do *software*. Eles podem, então, propor novos requisitos de sistema.

- Validação de requisitos - O protótipo pode revelar erros e omissões nos requisitos propostos. Uma função descrita em uma especificação pode parecer útil e bem-definida. Contudo, quando essa função é utilizada com outras, os usuários muitas vezes acham que sua visão inicial era incorreta e incompleta. A especificação de sistema pode então ser modificada para refletir sua compreensão alterada dos requisitos.

Para De Souza et al (1999), um protótipo é uma aplicação, normalmente experimental e incompleta, que permite aos *designers* avaliarem suas ideias de projeto durante o processo de criação da aplicação pretendida. Ele deve ser construído rapidamente e com baixo custo e seu tempo de vida não é definido. Dentre as informações extraídas de um protótipo, pode-se destacar a funcionalidade necessária ao sistema, sequências de operação, necessidades de suporte ao usuário, representações necessárias e comunicabilidade da aplicação (PREECE et al, 1994).

Há protótipos de baixa fidelidade, como também de alta fidelidade (PREECE, ROGERS e SHARP, 2005). A primeira categoria é útil já que tende a ser simples, de baixo custo e rápida de produção. Pode ser ainda rapidamente modificada, oferecendo, portanto, suporte à exploração de *designs* e ideias alternativas. A prototipação de alta fidelidade, no entanto, utiliza materiais que se espera que estejam no produto final, levando assim, mais tempo para serem construídos. Na prática, diferentes protótipos e seus níveis associados de fidelidade podem fornecer tipos diversos de informação de *design*.

Em relação ao seu objetivo de avaliação, a prototipação pode ser classificada como exploratória, experimental e evolutiva. A prototipação exploratória é usada para ajudar a esclarecer requisitos dos usuários, ou para examinar uma variedade de opções de solução de *design* para que se determine a mais adequada. A prototipação experimental enfatiza aspectos técnicos do desenvolvimento, oferecendo aos desenvolvedores resultados experimentais para a tomada de decisões de *design* e implementação. Finalmente, a prototipação evolutiva avalia o impacto que a introdução de novas tecnologias pode trazer para uma pessoa, seu modo de trabalhar e para a organização como um todo. Neste caso, os *designers* devem trabalhar em cooperação com os usuários em um processo contínuo de reengenharia (DE SOUZA et al, 1999).

De acordo com De Souza et al (1999), uma das grandes vantagens de protótipos são eles serem parte de um *design* iterativo centrado no usuário. Isso permite que os *designers* experimentem ideias junto a usuários e recebam seu *feedback*.

A prototipação é, de modo análogo, uma maquete para a arquitetura, de um sistema futuro com o qual pode-se realizar verificações e experimentações para se avaliar algumas de suas qualidades antes que o sistema venha realmente a ser construído (PREECE, ROGERS e SHARP, 2005).

2.6.4 UML

A *Unified Modeling Language* (UML), que em português fica, Linguagem de Modelagem Unificada, foi desenvolvida por *Grady Booch*, *James Rumbaugh*, e *Ivar Jacobson* que são conhecidos como "os três amigos". Eles possuem um extenso conhecimento na área de modelagem orientada a objetos já que as três mais conceituadas metodologias de modelagem orientada a objetos foram eles que desenvolveram e a UML é a junção do que havia de melhor nestas três metodologias adicionado novos conceitos e visões da linguagem (MEDEIROS, 2004).

Modelos orientados a objetos são implementados convenientemente utilizando uma linguagem de programação orientada a objetos. A engenharia de *software* orientada a objetos é muito mais que utilizar mecanismos de sua linguagem de programação, é saber utilizar da melhor forma possível todas as técnicas da modelagem orientada a objetos. Ela está totalmente baseada em conceitos e testes provenientes das metodologias existentes anteriormente, e também é muito bem documentada com toda a especificação da semântica da linguagem representada em meta-modelos (RUMBAUGH, 1999).

Uma classe é a descrição de um tipo de objeto. Todos os objetos são instâncias de classes, onde a classe descreve as propriedades e comportamentos daquele objeto. Objetos só podem ser instanciados de classes. Usam-se classes para classificar os objetos que são identificados no mundo real (RUMBAUGH, 1999).

A UML trabalha com 10 diagramas principais, conforme Figura 14. Para uso deste trabalho foram utilizados os Diagramas de Classe, Caso de Uso e de Sequência. Este

último está inserido no Diagrama de Interação (Diagrama de Sequência, Diagrama Geral de Interação, de Comunicação e de Tempo).

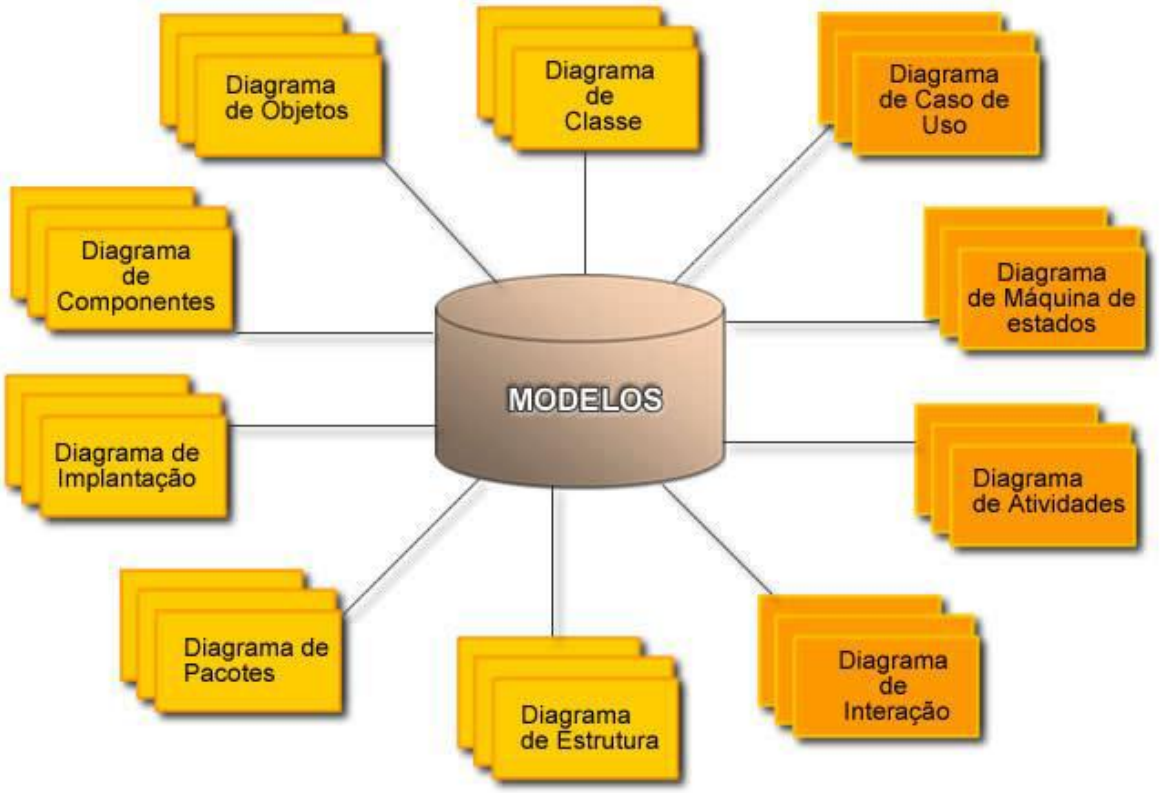


Figura 14: Tipos de diagramas UML (INFOESCOLA, 2013).

2.6.5 Banco de Dados

Banco de dados (BD) é um conjunto de dados persistentes, ou seja, são coleções de informações que se relacionam de forma a criar um sentido, com o intuito de armazenar informações de uma determinada organização. São de vital importância para empresas, e há duas décadas se tornaram a principal peça dos sistemas de informação. Esses dados são mantidos por um *software* chamado de Sistema Gerenciador de Banco de Dados (SGBD), onde os usuários desse sistema podem realizar busca, exclusão, inserção e alteração nesses arquivos de BD (DATE, 2003).

Alguns autores definem que dados e informações tem o mesmo significado. Por outro lado, outros definem dados como os valores fisicamente armazenados no banco de dados e informações como o significado gerado a partir de um determinado dado (DATE, 2003).

2.6.6 Projeto de Banco de Dados

Na maioria dos projetos de banco de dados são utilizados dois modelos de banco de dados em duas fases: o modelo conceitual e o modelo lógico, porém existe mais um modelo: o físico.

Na primeira fase é abstraído o modelo conceitual. Este modelo descreve o BD de forma independente, sem necessitar do SGBD. O projeto de BD é representado em um Diagrama de Entidade Relacionamento (DER), e através de regras heurísticas é possível construir modelos como mostra a Figura 15 (HEUSER, 1998).

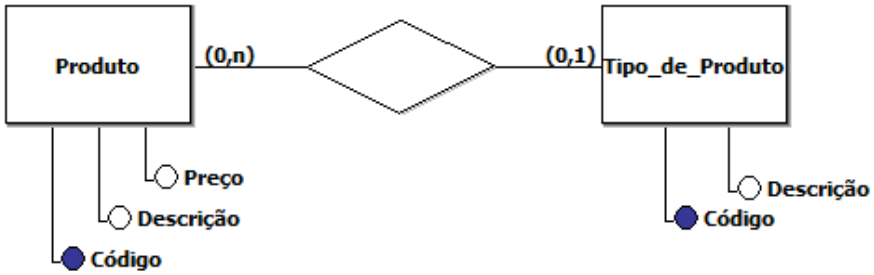


Figura 15: Projeto de Banco de Dados Conceitual.

E na segunda fase é abstraído o modelo lógico. Modelo que representa o BD na maneira como é visto pelo usuário do SGBD, ou seja, é uma descrição de um banco de dados no nível de abstração visto pelo usuário do SGBD. Assim, o modelo lógico é dependente do tipo particular de SGBD que está sendo usado. Ao contrário do conceitual, depende do SGBD utilizado no projeto. E é gerado a partir do modelo conceitual desenvolvido na primeira fase do projeto de BD (HEUSER, 1998), conforme ilustra a Figura 16.

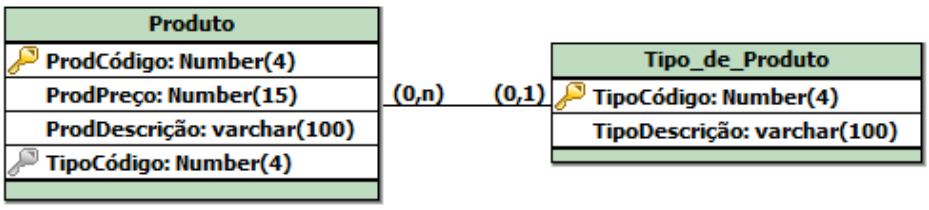


Figura 16: Exemplo de Banco de Dados Modelo Lógico (HEUSER, 1998).

2.6.7 A Linguagem SQL

A *Structured Query Language* (SQL), que traduzido para o português, significa Linguagem de Consulta Estruturada,ou seja, é a linguagem de pesquisa declarativa padrão para definição e manipulação no BD relacional (IBMc, 2011). É uma linguagem simples e de fácil uso (DAMAS, 2007).

A linguagem SQL se divide em três principais grupos: DDL, DML e DCL. A *Data Definition Language* (DDL) – Linguagem de Definição de Dados, trabalha com os objetos e tem os seguintes comandos: *ALTER* – altera um objeto do banco de dados (uma tabela, por exemplo), *CREATE* – cria um objeto na base de dados e *DROP* – apaga um objeto da base de dados; Os comandos *ALTER* e *CREATE,* podem ser usados para *index* (indices) e *view* (visões) (DAMAS, 2007).

Outro grupo é a *Data Manipulation Language (DML), ou seja,* – Linguagem de Manipulação de Dados, ela trabalha com as *tuplas* (linhas), seus comandos são *SELECT* – consulta os dados armazenados em uma tabela, *INSERT* – insere uma linha na tabela, *DELETE* – deleta e *UPDATE* – permite alterar quantas linhas de dados for preciso em uma tabela (DAMAS, 2007).

E, por último, porém não menos importante, a *Data Control Language* (DCL) – Linguagem de Controle de Dados) trabalha com os utilizadores, controla o acesso aos dados, seus principais comandos são: *GRANT* – seta os privilégios, permite o acesso aos dados ao usuário e *REVOKE* – remove os privilégios dado ao usuário (DAMAS, 2007).

A linguagem SQL faz parte de uma das cinco gerações de linguagens, a quarta geração. Ela atende a quase todas as necessidades para o desenvolvimento de um banco de dados, porém para completar as necessidades que a SQL não atende, em algumas ocasiões o desenvolvedor concilia a linguagem SQL com alguma outra linguagem de programação (DAMAS, 2007).

2.6.8 Orientação a Objetos

É uma abordagem para o desenvolvimento de *software* que organiza os problemas e suas soluções como um conjunto de objetos distintos. A estrutura e o comportamento dos dados estão incluídos na representação. Pode-se reconhecer uma representação OO por suas sete características: identidade, classificação, abstração, encapsulamento,

herança, polimorfismo e persistência. Dentre esses, veja alguns conceitos mais relevantes que precisam ser entendidos antes de se iniciar uma programação orientada a objetos, como classe, objetos, herança e polimorfismo (PRESSMAN, 1995).

2.6.9 Classes

Uma classe é um modelo ou protótipo onde os objetos serão criados e definidos. Assim, a classe define o estado e o comportamento de um objeto físico do mundo real, como está representado na Figura 17 (RICARTE, 2001).

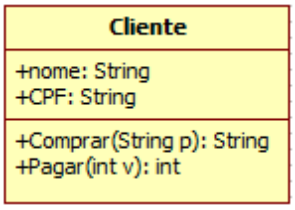


Figura 17: Exemplo de Classe.

2.6.10 Objetos

Objeto é uma instância de uma classe, sendo assim ele é um objeto físico do mundo real. São justamente os objetos que caracterizam a programação OO. O objeto tem seus devidos atributos e métodos que o manipulam (RICARTE, 2001).

Informalmente um objeto representa uma entidade, tanto física quanto conceitual ou de *software*. Exemplos:

" Entidade Física: caminhão, carro, bicicleta, etc.

" Entidade Conceitual: processo químico, matrícula, etc

" Entidade de *Software*: lista encadeada, arquivo, etc.

Podemos afirmar que um objeto é um conceito, abstração, ou entidade com limites bem definidos e um significado para a aplicação (RICARTE, 2001).

2.6.11 Polimorfismo

É a habilidade de variáveis terem “mais de um tipo”. Funções são ditas polimórficas quando seus operandos podem ter mais de um tipo (SIERRA, 2007).

Polimorfismo consiste em implementar um código considerando classes abstratas ou *interface*s, ao invés de classes concretas (RICARTE, 2001).

2.6.12 Herança

A herança é um mecanismo existente no paradigma OO, ela organiza e estrutura o *software*, permitindo a reutilização da estrutura e do comportamento de uma classe ao se definir novas classes. Assim as classes herdam o estado e o comportamento de suas superclasses. Com a herança, classes podem herdar características da classe pai, como por exemplo, atributos e métodos. Podendo assim a subclasse, especificar ou estender a superclasse (RICARTE, 2001).

A Figura 18 apresenta um exemplo de herança e polimorfismo. Foram criadas três classes de tipos diferentes de canetas, porem todas são derivadas de outra classe. As classes CanetaFlor, CanetaPalhaço e Caneta Relógio herdam os atributos e métodos (mesma assinatura) da classe Caneta. Porém, as ações dos métodos são diferentes para cada uma das classes que estão herdando (MARIANI, 2010).

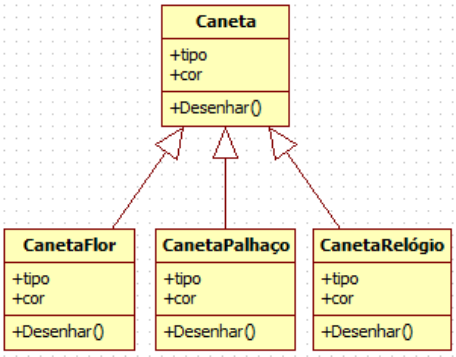


Figura 18: Representação de Herança e Polimorfismo.

2.6.13 Java

O Java possui várias tecnologias. Uma delas é a linguagem de programação Java criada pela *Sun* e mantida através de um comitê no Brasil. No mundo vários usuários se formaram para disseminar o conhecimento da linguagem Java e muitos grupos foram criados para auxiliarem no suporte. Um deles é o Grupo de Usuários Java (GUJ) uma comunidade virtual com artigos, tutoriais e fórum para tirar dúvidas, o maior em língua portuguesa com mais de cem mil usuários e um milhão de mensagens. As principais vantagens da utilização da linguagem de programação Java são: Portabilidade

multiplataforma; é um *software* de uso gratuito, bem como, os servidores e ambientes de aplicação; é possível a reusabilidade do código; o suporte no Brasil possui a maior comunidade virtual de Java do mundo (JAVA, 2014).

Java é uma linguagem de programação utilizada no desenvolvimento de *software* que tem portabilidade entre as plataformas, através da máquina virtual ou *Java Virtual Machine* (JVM). Além disso, ela é OO, o que possibilita a reutilização de códigos desenvolvidos em outros projetos e trabalha com a troca de mensagens entre os objetos (JAVA, 2014). Uma delas que é bem conhecida, é o *software* Eclipse, representando o ambiente *Java Development Kit* (JDK) para desenvolvimento em aplicações para *desktop* e *web*, disponibilizado pela própria *Sun Microsystem*, empresa de criação do Java. Outro ambiente de desenvolvimento em Java é o *Software Development Kit* (SDK). Este não possui dependência direta com o sistema operacional subjacente, é disponibilizado por qualquer empresa ou projetos *opensource*. A dependência principal é, portanto, sobre a plataforma Java em SI (DEITEL, 2006).

2.7 FERRAMENTAS

2.7.1 ASTAH COMMUNITY

A ferramenta *Astah Community* é *open source* e é utilizada para o desenvolvimento da modelagem de *software*. É flexível e extensível e contém vários recursos. Nela é possível desenvolver vários diagramas: diagrama de casos de uso, diagrama de classe, diagrama de sequência, diagrama de estados, diagrama de atividades, diagrama de componentes, diagrama de implantação, diagrama de estrutura composta, diagrama de comunicação e diagrama de pacote (ASTAH_COMMUNITY, 2014).

2.7.2 NETBEANS

A ferramenta NetBeans foi fundada no ano de 2000 pela Sun *Microsystems*, totalmente *open-source*. É um ambiente de desenvolvimento integrado para desenvolvedores de *software*. Escrito em totalmente em Java, porém compila qualquer tipo de linguagem (NETBEANS, 2011). O IDE vem com *drivers* para os servidores de banco de dados MySQL e PostgreSQL (NETBEANS, 2014).

2.7.3 MYSQL

O MySql é o SGBD de código aberto mais conhecido no mundo que trabalha com a linguagem SQL. Pois tem excelente desempenho, confiabilidade e facilidade de uso. Tem uma grande portabilidade, funciona em mais de 20 plataformas, sendo elas as principais, *Linux, Windows, Mac e Solaris* e foi escrito em C e C++ (MYSQL, 2014).

2.7.4 XMind

XMind é uma ferramenta de código aberto para a construção de fluxograma no formato de mapas mentais e *brainstorming* onde o usuário poderá acrescentar relacionamentos, limites, sumários, observações, marcadores e muito mais. Os mapas gerados podem ser exportados para diversos formatos, e a ferramenta destina-se principalmente a ajudar os usuários a capturar ideias, a organizar vários gráficos, e a compartilhar de maneira colaborativa o conhecimento armazenado.

Uma grande vantagem do *Xmind* é ser multiplataforma disponível para *Windows*, Mac OS X e Linux, existindo também uma versão portátil, ou seja, é possível rodar diretamente do *pen-drive* (XMIND, 2014).

3 METODOLOGIA

3.1 O AMBIENTE DO ESTUDO

Este estudo foi realizado no âmbito do MS/Departamento de Informática do Sistema Único de Saúde (DATASUS) e Universidade de Brasília (UnB), no qual teve extensão no Laboratório de Informática em Saúde (LIS) localizado na UnB *Campus* Faculdade Gama (FGA) e no Laboratório de Arquitetura Orientado a Serviço (LabSOA), este especializado em pesquisa e desenvolvimento para *Service-Oriented Architecture* (SOA) localizado no Instituto Brasília de Tecnologia e Inovação (IBTI).

Para a realização dos estudos e da pesquisa, os laboratórios ofereceram apoio de professores doutores, analistas e desenvolvedores de sistemas, técnicos especializados e alunos mestrandos e graduandos de iniciação científica.

3.2 DELIMITAÇÃO DO ESTUDO

A primeira fase do desenvolvimento dessa pesquisa consiste no estudo sistematizado do domínio. Nessa etapa são realizadas consultas aos especialistas da área e estudo de casos relacionados ao tema, além da revisão da literatura a fim de investigar a viabilidade e a relevância da pesquisa.

O objetivo da segunda fase é desenvolver, por meio do processo de engenharia de *software*, a Prototipação. Nessa etapa é possível ter uma visão inicial do sistema de *software* proposto, através da documentação contextual, glossário de termos e descrição dos envolvidos.

Em seguida é realizado o levantamento dos requisitos funcionais e não funcionais, onde é possível ter o espectro das tarefas e técnicas que levam ao entendimento da proposta e, em geral, para conhecer o problema e suas possíveis soluções.

A etapa seguinte trata da delimitação do escopo do trabalho, onde de acordo com a complexidade do SIS do sumário de alta hospitalar, é possível visualizar a importância de se fazer a delimitação do escopo, considerando o tempo necessário para se desenvolver um modelo de conhecimento que dê conta de representar toda essa

complexidade, assim como também o principal objetivo é se atingir a renderização de arquétipo do contexto do sumário de alta hospitalar e representá-la em tela.

Em suma, a ideia da prototipação é demonstrar o processo que possibilita que o programador de *software* crie um modelo que pode vir ser construído.

A modelagem do protótipo é realizada com a utilização das ferramentas *Astah Community* (versão 6.0.2, 2014) e o *XMind* 2013 (versão 3.4.1), para proporcionar um padrão para a arquitetura do protótipo de *software*. A linguagem que foi usada para a implementação é o Java (versão 1.7, 2013) e a NetBeansIDE (versão 8, 2014), bem como foi selecionado um SGBD livre, o MySQL (versão 5.1, 2014).

4 ESTUDO DE CASO

4.1 ESTUDO DE CASO: SUMÁRIO DE ALTA HOSPITALAR

4.1.1 Definição do Estudo de Caso

O estudo de caso em Sumário de Alta, baseado no modelo de informação (ANEXO 1) e no modelo de arquétipos, ocorreu no primeiro semestre de 2014 no âmbito do projeto de pesquisa RES Nacional, Arquitetura Corporativa do Cartão Nacional de Saúde (CNS) e Centro de Excelência em SOA, através da parceria da UnB e MS.

4.1.2 Documento Contextual do Protótipo

O documento contextual do protótipo se concentra principalmente em criar uma visão macro do negócio do sistema. A partir deste documento já é possível se ter uma ideia do sistema e seu enquadramento no ambiente ao qual vai servir, buscando-se prever a viabilidade de construção do projeto, conforme a Figura 19.

Descrição Inicial	O protótipo de sistema a ser desenvolvimento deve viabilizar a transparência e a interoperabilidade das informações contidas nos arquétipos representados no contexto do Sumário de Alta Hospitalar aplicados a usuários de SIS dentro de soluções propostas no projeto RES/SUS Brasil, para estudo de caso aplicado do RES Clínico V1.
Problema Original	Dentro do contexto do projeto RES/SUS Brasil, existe a necessidade de renderizar os arquétipos definidos do Sumário de Alta Hospitalar, ou seja, demonstrar automaticamente as informações contidas nos arquétipos em forma de *interface* gráfica.
Origem do Sistema	Atualmente não existe nenhum sistema formal de intercâmbio das informações contidas nos arquétipos para uma *interface* gráfica, uma vez que o modelo de informação do Sumário de Alta Hospitalar ainda é um projeto de norma.
Contexto de	O protótipo de sistema será restrito a membros do projeto RES/SUS Brasil para estudo de caso aplicado do RES Clínico

Utilização	V1 e para fins acadêmicos.
Limites do Protótipo	O protótipo de sistema deverá efetuar identificação ou novo registro de paciente, verificar características, procedimentos e regime de internação, prescrever orientações ao paciente e imprimir histórico de alta do paciente.

Figura 19 : Caso Sumário de Alta Hospitalar – Documento Contextual do Protótipo.

Obedecendo as regras de modelagem de dados e criação de diagramas da UML, é ilustrado nas Figuras 20 e 21 o diagrama de Caso de Uso e o Diagrama de Classe, respectivamente.

O Diagrama de Caso de Uso foi utilizado para representar as ações de eventos dos atores envolvidos com o sistema. Neste caso, os atores são os profissionais da saúde, ou seja, as pessoas que interagem com o sistema, representados por bonecos, conforme é ilustrado na Figura 20. Esses três profissionais são as pessoas que utilizam o sistema, possuindo acesso através de um *login* e uma senha, devidamente cadastrado pelo desenvolvedor do sistema.

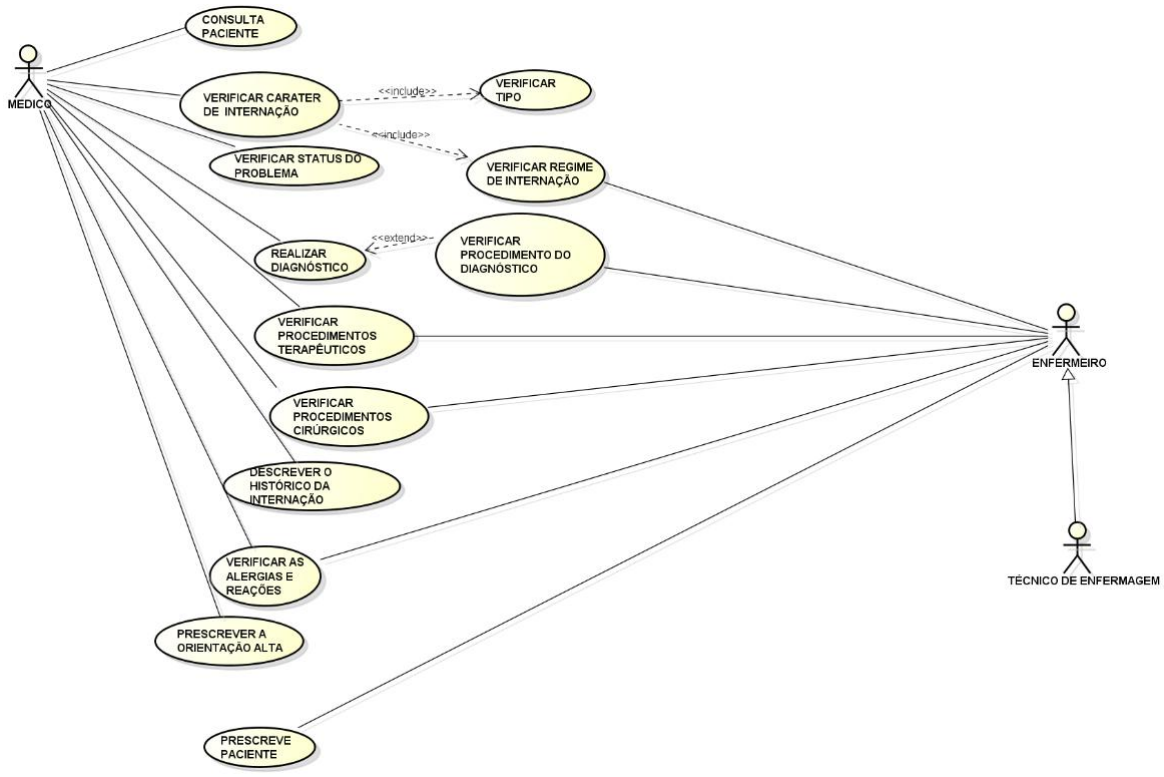


Figura 20: Diagrama de Caso de Uso do Sumário de Alta Hospitalar.

O diagrama de classe foi criado para auxiliar na visualização das ações da programação OO em Java. Cada classe é dividida em 3 partes, a 1ª é o nome da classe criada, a 2ª representa os atributos utilizados com o tipo de dados explicitados e a última divisão representa as ações de cada classe para implementar uma operação do sistema (Figura 21).

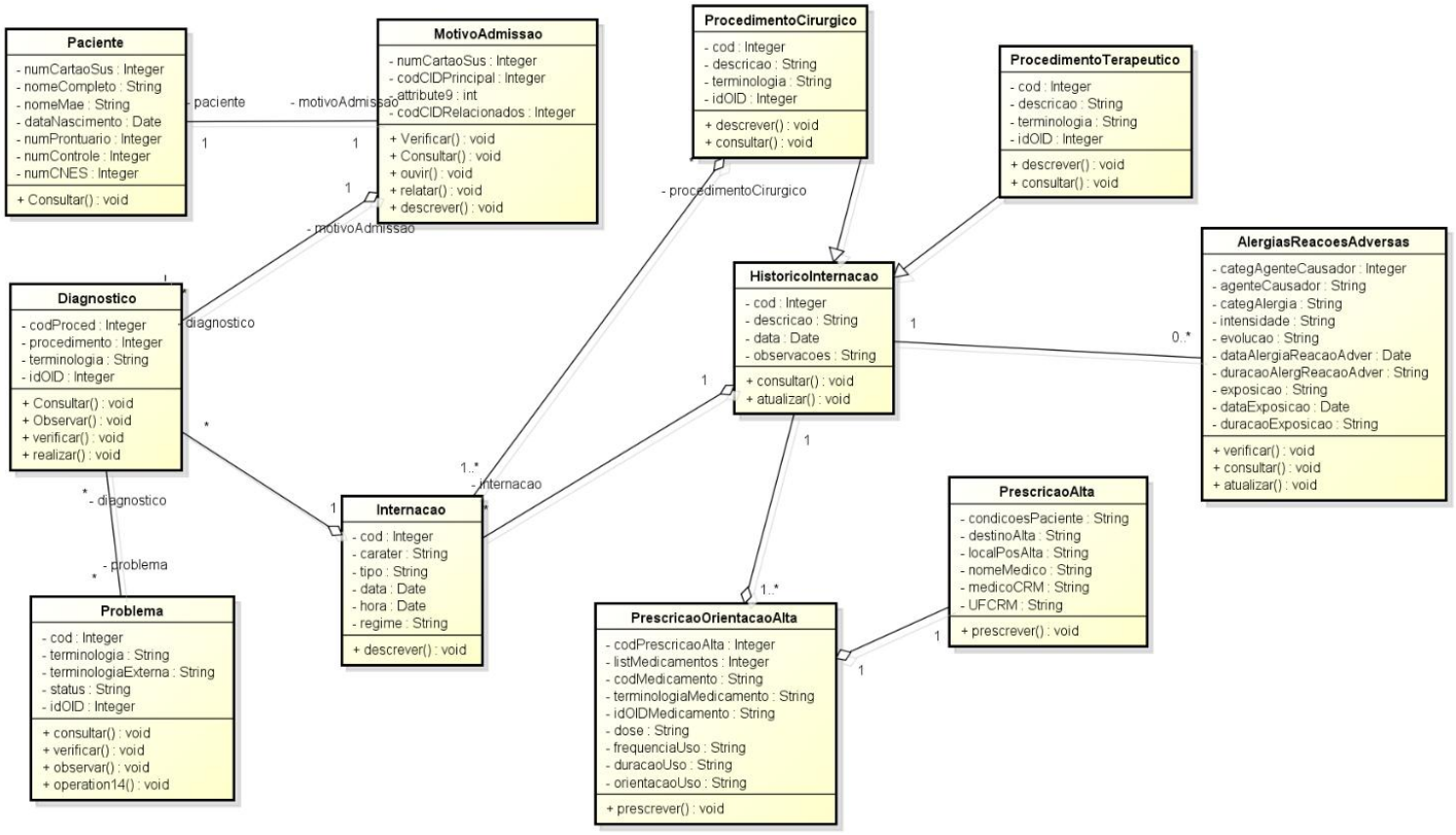


Figura 21: Diagrama de Classe do Sumário de Alta

4.1.3 Glossário de Termos e Descrição dos Envolvidos

O glossário de termos serve para documentar informações que requeiram melhores esclarecimentos, de modo a melhorar a comunicação e evitar mal-entendidos. Possui o caráter de registrar não apenas dados, mas também quaisquer informações e restrições que não possam ser colocadas nos diagramas construídos ao longo do processo de análise e projeto de sistemas (PREECE, ROGERS e SHARP, 2005). Os termos podem ser vistos na Figura 22.

Termo	Categoria	Descrição
Administrador	Ator	Terá como função cadastrar, identificar paciente cadastrado e imprimir resumo de alta do paciente.
Médico	Ator	Este é um ator que poderá efetuar consulta médica no paciente, realizar diagnóstico, verificar histórico e regime de internação, caráter de internação, status do problema, verificar procedimentos terapêuticos, cirúrgicos e diagnósticos, verificar as alergias e reações adversas, prescrever o paciente e prescrever orientações de alta.
Enfermeiro	Ator	Este ator poderá verificar regime de internação, verificar os procedimentos diagnósticos, terapêuticos e cirúrgicos, verificar alergias e reações adversas e prescrever paciente.
Técnico em Enfermagem	Ator	Este ator poderá verificar regime de internação, verificar os procedimentos diagnósticos, terapêuticos e cirúrgicos, verificar alergias e reações adversas e prescrever paciente.
Administrador	Ator de Negócio	Terá como função cadastrar paciente, e poderá identificar paciente cadastrado. Este ator é considerado ator de negócio para fins de uma nova forma de representação e função dentro do sistema.

Médico	Ator de Negócio	Este é um ator que poderá efetuar consulta médica no paciente, realizar diagnóstico, verificar histórico e regime de internação, caráter de internação, status do problema, verificar procedimentos terapêuticos, cirúrgicos e diagnósticos, verificar as alergias e reações adversas, prescrever o paciente e prescrever orientações de alta. Este ator é considerado ator de negócio para fins de uma nova forma de representação e função dentro do sistema.
Enfermeiro	Ator de Negócio	Este ator poderá verificar regime de internação, verificar os procedimentos diagnósticos, terapêuticos e cirúrgicos, verificar alergias e reações adversas e prescrever paciente. Este ator é considerado ator de negócio para fins de uma nova forma de representação e função dentro do sistema.
Técnico em Enfermagem	Ator de Negócio	Este ator poderá verificar regime de internação, verificar os procedimentos diagnósticos, terapêuticos e cirúrgicos, verificar alergias e reações adversas e prescrever paciente. Este ator é considerado ator de negócio para fins de uma nova forma de representação e função dentro do sistema.
Viabilizar o controle do paciente no estabelecimento de saúde	Objetivo	Objetivo do protótipo de sistema
Identificar nº cartão do SUS ou nome	Caso de uso	Caso de uso expandido do objetivo

Figura 22: Glossário de Termos e identificação do paciente.

4.1.4 Levantamento de Requisitos

Nesta seção apresentam-se os requisitos funcionais e não funcionais do sistema, conforme as Tabelas 1 e 2.

Tabela 1: Requisitos Funcionais.

Código	**Requisitos Funcionais**
RF01	Possuir 4 tipos de usuários: Administrador, Médico, Enfermeiro e Técnico em Enfermagem
RF02	Para o usuário Administrador permitir o gerenciamento de: RF0201: Cadastrar paciente RF0202: Identificar paciente
RF03	Para o usuário Médico permitir o gerenciamento de: RF0301: Descrever consulta; RF0302: Verificar histórico de internação; RF0303: Verificar caráter de internação; RF0304: Verificar status do problema; RF0305: Verificar procedimentos terapêuticos realizados; RF0306: Verificar procedimentos cirúrgicos realizados; RF0307: Verificar procedimentos de diagnósticos realizados; RF0308: Verificar as alergias e reações adversas do paciente; RF0309: Prescrever o paciente; RF0310: Prescrever orientações de alta;
RF04	Para o usuário Enfermeiro permitir o gerenciamento de: RF0401: Verificar regime de internação; RF0402: Verificar os procedimentos diagnósticos realizados; RF0403: Verificar procedimentos terapêuticos realizados; RF0404: Verificar procedimentos cirúrgicos realizados; RF0405: Verificar alergias e reações adversas;

	RF0406: Prescrever paciente;
RF05	Para o usuário Técnico em Enfermagem permitir o gerenciamento de: RF0501: Verificar regime de internação; RF0502: Verificar os procedimentos diagnósticos realizados; RF0503: Verificar procedimentos terapêuticos realizados; RF0504: Verificar procedimentos cirúrgicos realizados; RF0505: Verificar alergias e reações adversas; RF0506: Prescrever paciente.
RF06	RF0601: Gerar relatório a partir do histórico de alta do paciente.

Tabela 2: Requisitos Não Funcionais.

Código	**Requisito Não Funcional**
RNF 01	Acessível apenas em plataforma web.
RNF 02	Desenvolvido em Java.
RNF 03	Compatível com *Windows* e *Linux*.
RNF 04	MSQL – Sistema SGBD utilizado.

4.2 DELIMITAÇÃO DO ESCOPO DE TRABALHO

4.2.1 Delimitação do Escopo

Observando a Figura 12, que ilustra a complexidade do SIS do Sumário de Alta Hospitalar, é importante neste ponto fazer uma delimitação do escopo de trabalho, considerando essencialmente o tempo necessário para se desenvolver um protótipo de *software* que dê conta de representar toda essa complexidade.

Optou-se, então, por fazer um estudo de caso em um subsistema do modelo de informação do sumário de alta hospitalar e, portanto, cabe ressaltar que esse subsistema (Dados do Paciente) desenvolvido deva atender parcialmente e não todo um SIS, apesar de ter aderência ao modelo completo pelo uso de conceitos comuns e pela adoção do padrão openEHR como modelo de referência.

4.2.2 Definição do Arquétipo

De acordo com a ontologia de conceitos, as informações de identificação do paciente, ou seja, informações demográficas do paciente, podem-se destacar no sumário de alta, a representação das informações básicas dos pacientes, tais como: nome, identificador, sexo, idade, registro do paciente, ocupação e origem. Estas informações podem ser modeladas utilizando-se o arquétipo denominado openEHR-DEMOGRAPHIC-PERSON.person_patient.v1, encontrado no repositório de arquétipos *Clinical Knowledge Manager* (CKM)(OPENEHR, 2014c) e com tradução para o português.

Como estratégia para a diagramação da informação, utiliza-se a representação por "Mapa Mental" que corresponde ao objetivo de negócio identificado no parágrafo anterior, com a finalidade de conectar conceitos adjacentes na ideia central do arquétipo (Figura 23).

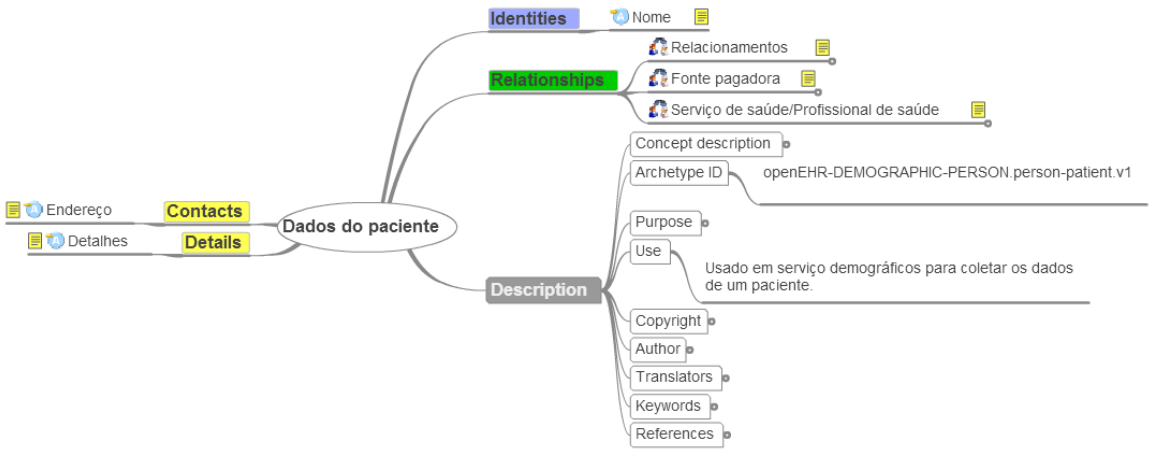


Figura 23: Diagrama do arquétipo openEHR-DEMOGRAPHIC-PERSON.person_patient.v1(Dados do paciente).

A representação do arquétipo openEHR-DEMOGRAPHIC-PERSON.person_patient.v1 (Dados do paciente) foi feita através da linguagem ADL (ANEXO 2).

4.2.3 Validação do Arquétipo definido

O *parse* possui uma *interface* de linha de comando construído principalmente para fornecer uma maneira interativa para validar arquétipos. Para isso basta incluir todas as bibliotecas dependentes e digitar o seguinte comando:

java se.acode.openehr.parser.ADLParser [adlFile] (OPENEHR, 2014d).

Para execução do parse proposto nesse trabalho pode-se identificar a validação do arquétipo através das bibliotecas inclusas e o comando principal, conforme a Figura 24.

```
import org.openehr.am.archetype.Archetype;

import org.openehr.am.archetype.constraintmodel.ArchetypeSlot;

import org.openehr.am.archetype.constraintmodel.CAttribute;

import org.openehr.am.archetype.constraintmodel.CComplexObject;

import org.openehr.am.archetype.constraintmodel.CObject;

import org.openehr.am.archetype.ontology.*;

import se.acode.openehr.parser.ADLParser;
```

Figura 24: Validação do Arquétipo definido.

4.2.4 A Modelagem da proposta

Essa etapa permite demonstrar o arquétipo representado pela linguagem de programação ADL, através do *parse* utilizando como base o modelo de referência openEHR, conforme o código mostrado na Figura 25.

```
package br.unb.meb.openehr.adl;

import org.openehr.am.archetype.Archetype;

import org.openehr.am.archetype.constraintmodel.ArchetypeSlot;

import org.openehr.am.archetype.constraintmodel.CAttribute;

import org.openehr.am.archetype.constraintmodel.CComplexObject;

import org.openehr.am.archetype.constraintmodel.CObject;

import org.openehr.am.archetype.ontology.*;

import se.acode.openehr.parser.ADLParser;
```

```
import java.io.File;

import java.text.ParseException;

import java.util.List;

public class MainApp {

   public static void main(String[] args) {

            String file_demographic_person =
"/home/alfredo/GoogleDrive/Mestrado/Projetos/openehr-adl-
parser/src/main/resources/openEHR-DEMOGRAPHIC-PERSON.person-patient.v1.adl"

      String file_str = file_demographic_person;

      File file = new File(file_str);

      ADLParser parser = null;

      Archetype archetype = null;

      try {
```

```
         parser = new ADLParser(file, true, true);

         archetype = parser.parse();

         CComplexObject adlObject = archetype.getDefinition();

         List<CAttribute> adlAttributes = adlObject.getAttributes();

         ArchetypeOntology ontology = archetype.getOntology();

         if (adlAttributes != null) {

            int i = 0;

            for (CAttribute attribute : adlAttributes) {
```

```
                System.out.println("Index i: " + i + " [RmAttributeName]: " +
attribute.getRmAttributeName());

                List<CObject> children = attribute.getChildren();

                if (children != null) {

                   int j = 0;

                   for (CObject child : children) {

                      String nodeId = child.getNodeID();

                      System.out.println("Index j: " + i + " [NodeID]: " + nodeId + " Text: "
+ ontology.termDefinition("pt-br", nodeId).getText());

                      ArchetypeSlot slot = (ArchetypeSlot) child;

                   }

                }

                i++;

             }

          }

          List<String> terminologies = ontology.getTerminologies();

          List<OntologyDefinitions> definitions = ontology.getTermDefinitionsList();

          if (definitions != null) {

             for (OntologyDefinitions definition : definitions) {

                String lang = definition.getLanguage();

                System.out.println("Language: " + lang);

                List<ArchetypeTerm> terms = definition.getDefinitions();
```

```
                if (terms != null) {
                    for (ArchetypeTerm term : terms) {
                        System.out.println(lang + " :: code :: " + term.getCode() + " :: Text :: "
+ term.getText());
                    }
                }
            }
        }
    } catch (ParseException e) {
        e.printStackTrace();
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        if (parser != null) {
            parser = null;
        }
        if (archetype != null) {
            archetype = null;
        }
    }
  }
}
```

Figura 25: Execução do *parser* ADL Java no arquétipo openEHR-DEMOGRAPHIC-PERSON.person-patient.v1(Dados do paciente).

4.2.5 Guia de execução do parser ADL Java

Pode se inicializar o construtor de um *parse* de três formas, isto é, tanto com um arquivo ADL, com uma *String* que contém o arquétipo em formato ADL, ou carregado a partir de um BD (OPENEHR, 2014a). Exemplos:

- Arquivo adlFile = new File ("my_archetype.adl");
- ADLParser parser = new ADLParser (adlFile);
- Cordas adlText = ...; Provavelmente carregado a partir do banco de dados do analisador ADLParser = novo ADLParser (adlText).

No estudo de caso desse trabalho, o construtor do *parser* se inicia com o comando da Figura 26.

```
String file_demographic_person =
"/home/alfredo/GoogleDrive/Mestrado/Projetos/openehr-adl-
parser/src/main/resources/openEHR-DEMOGRAPHIC-PERSON.person-
patient.v1.adl";

String file_str = file_demographic_person;

    File file = new File (file_str);
```

Figura 26: Comandos de inicialização da instância do *parse* do arquétipo definido.

O próximo passo será a chamada *parser.parse* () para obter a instância Arquétipo, com o seguinte código da Figura 27.

```
ADLParser parser = null;

    Archetype archetype = null;

    try {
```

```
parser = new ADLParser (file, true, true);

archetype = parser.parse ();
```

Figura 27: Comando para se obter a instância do arquétipo definido.

Nessa etapa temos um objeto chamado *archetype*. Dentro desse *archetype*, tem-se um método chamado de *getDefinition* (*archetype.getDefinition* ()), que dá a área do arquétipo, ou seja, a definição da área, sendo essa área chamada de *CComplexObject* (*CComplexObject adlObject*). Em outras palavras, um objeto complexo dentro da sua estrutura de classe. Dessa forma, o objeto complexo *CComplexObject adlObject = archetype.getDefinition* (), sendo que o *CComplexObject* sempre terá um atributo chamado de *getAttributes* (*adlAttributes = adlObject.getAttributes*()).

Os atributos estão contidos na lista (List<CAttribute> adlAttributes = adlObject.getAttributes()) dos objetos complexos (CComplexObject). Cada atributo dessa lista é uma subsessão da definição, ou seja, para o arquétipo em estudo é identificamos 5 atributos, os quais são: *description* (identificação, descrição, representação, uso, autor, ano, referência e linguagem), *details* (detalhes demográficos do paciente), *identities* (conjunto de dados que especificam o nome do paciente), *contacts* (endereços vinculados a um único contato) *e relacionships* (relacionamento do paciente, especialmente laços familiares, com fontes pagadores e com uma organização prestadora de serviços de saúde), sendo essa visível na figura 23.

Cada um desses atributos possuem seus *children*, que são filhos, ou seja, uma outra lista de objetos, chamada CObject, e esses *children* indicam através do códigos de termos [atxxxx] o termo *nodeId* ([NodeID]), de acordo com a Figura 28.

A seção *ontology* do arquétipo no formato ADL, encontra-se a identificação de cada termo,seguido da definição de seu atributo, conforme o código do Anexo 2.

```
if (children != null) {

        int j = 0;

        for (CObject child : children) {
```

```
                    String nodeId = child.getNodeID();

                    System.out.println ("Index j: " + i + " [NodeID]: " + nodeId + "
Text: " + ontology.termDefinition("pt-br", nodeId).getText());

                    ArchetypeSlot slot = (ArchetypeSlot) child;
```

Figura 28: Identificação dos atributos e atributos *children*.

Em continuação ao código, tem-se as definições buscadas dentro da seção *ontology*, ou seja, os códigos dos objetos presentes no arquétipo referentes a terminologias são buscados pelo método *ontology.getTerminologies*(), onde se encontra todos os atributos contidos na lista (*List<String> terminologies = ontology.getTerminologies*()). Da mesma forma acontece com o método: *definition.getLanguage*(), que através do código: *String lang = definition.getLanguage*(), encontra as traduções de outras linguagens, sendo o método *ontology.getTermDefinitionsList*() obrigatório para que se possa definir cada tradução realizada. Dessa forma, tem-se toda parte do código da seção *ontology* para esse arquétipo em estudo (Figura 29).

```
    List<String> terminologies = ontology.getTerminologies();

            List<OntologyDefinitions> definitions =
ontology.getTermDefinitionsList();

            if (definitions != null) {

                for (OntologyDefinitions definition : definitions) {

                    String lang = definition.getLanguage();

                    System.out.println("Language: " + lang);

                    List<ArchetypeTerm> terms = definition.getDefinitions();

                    if (terms != null) {

                        for (ArchetypeTerm term : terms) {
```

```
                    System.out.println(lang + " :: code :: " + term.getCode() + " ::
Text :: " + term.getText());
```

Figura 29: Código da seção *ontology* do arquétipo definido.

4.2.6 Pesquisa de Arquétipos de Referência no Arquétipo Raiz

O mapeamento entre os conceitos do arquétipo raiz “Dados do Paciente” (openEHR-DEMOGRAPHIC-PERSON.person-patient.v1) e os respectivos arquétipos são representados da seguinte forma:

- Os arquétipos “Nome do profissional de saúde” (openEHR-DEMOGRAPHIC-PARTY_IDENTITY.person_name-individual_provider.v1), “Estrutura do nome de uma pessoa.” (openEHR-DEMOGRAPHIC-PARTY_IDENTITY.person_name.v1), “Detalhes do nome pessoal de uma pessoa” (openEHR-HER CLUSTER.person_name.v1) e “Nome pessoal compatível com padrão Europeu”(openEHR-EHR-CLUSTER.person_name_isa.v1.). Estes possuem o conteúdo do termo "Conjunto de dados que especificam o nome da pessoa." [at0002] e o termo "Conjunto de dados que especificam o nome do paciente."[at0002.1]. Estes foram inseridos no *slot* do arquétipo “Dados do paciente” (openEHR-DEMOGRAPHIC-PERSON.person-patient.v1) e expressos na aplicação no campo “Nome”.

- Os arquétipos “Endereço” (openEHR-DEMOGRAPHIC-ADDRESS.address.v1), “Componentes de alto nível do endereço utilizados no Brasil” (openEHR-DEMOGRAPHIC-CLUSTER.high_level_address_other_data_br.v1), “Para registrar detalhes de um ou mais endereços pessoais.” (openEHR-EHR-CLUSTER.address.v1) e “Endereço compatíveis com Europeia Norma ISA.” (openEHR-EHR-CLUSTER.address_isa.v1), os quais possuem o conteúdo “Endereço” do termo [at0030]. Estes foram inseridos no *slot* do arquétipo “Dados do paciente” (openEHR-DEMOGRAPHIC-PERSON.person-patient.v1) e expressos na aplicação no campo “Endereço”.

- Os arquétipos “Identificador de um prestador de assistência à saúde.” (openEHR-DEMOGRAPHIC-CLUSTER.person_identifier-provider.v1) e “Identificador para uma pessoa.” (openEHR-DEMOGRAPHIC-CLUSTER.person_identifier.v1). Estes possuem o conteúdo “Identificações do beneficiário” do termo [at0.20] e “Identificação do beneficiário” do termo [at0.21] e foram inseridos no *slot* do arquétipo “Dados do paciente” (openEHR-DEMOGRAPHIC-PERSON.person-patient.v1) e expressos na aplicação no campo “Identificação do Beneficiário”.

- Os arquétipos “Identificador de um prestador de assistência à saúde.” (openEHR-DEMOGRAPHIC-CLUSTER.person_identifier-provider.v1) e “Identificador para uma pessoa.” (openEHR-DEMOGRAPHIC-CLUSTER.person_identifier.v1). Estes possuem o conteúdo “Identificações no prestador” do termo[at0.30] e “Identificação no prestador” do termo [at0.31] e foram inseridos no *slot* do arquétipo “Dados do paciente” (openEHR-DEMOGRAPHIC-PERSON.person-patient.v1), bem como expressos na aplicação no campo “Identificação no Prestador”.

4.2.7 Renderização de Arquétipos em Tela

Não foi possível criar a representação em tela do arquétipo definido “Dados do paciente” (openEHR-DEMOGRAPHIC-PERSON.person-patient.v1). O resultado dessa *interface*, foi devido a descoberta que não se pode fazer algo significativo (*interface* gráfica), se um dado arquétipo faz referência a outro arquétipo (através do uso de *Slots*). Assim essa referência mais cedo ou mais tarde terá de ser seguida para criar GUI ou preencher algum compartimento da estrutura de dados da aplicação.

5 DISCUSSÃO E CONCLUSÃO

O processo se inicia com a escolha do arquétipo, seja ele apenas um ou mais de um, a partir daí ao iniciar o processo de execução do *parser,* logo se aplica o método *getDefinition.* Esse método traz um objeto complexo (*CComplexObject adlObject*) e esse possui vários atributos como: *contacts, identities, details, relacionships e description,* sendo que cada atributo possui seus *children,* ou seja, seus filhos.

A partir desse ponto, o fato desse arquétipo escolhido ser do tipo *ArchetypeSlot*, ou seja, ele já traz a prospecção que outros arquétipos podem ou devem ser inclusos no momento da sua composição. Assim, possibilitou que essa descoberta influenciasse no fato de não saber que tipo os filhos dos atributos renderizariam, pois poderia ser tanto campos para criação de GUI, ou mesmo a inclusão de outro arquétipo.

A questão de interpretar e seguir com os Arquétipos *Slots*, traz a necessidade de se usar as referências de forma cíclicas, ou seja, de uma forma mais dinâmica, recebendo o MR com o AOM, construindo a GUI e, finalmente, a execução e o preenchimento das consultas. Para que isso acontecesse teria que se estudar a possibilidade de preencher o *slot* de uma forma mais explícita por um dos arquétipos que esteja determinado para inclusão de critérios, isso devido os arquétipos disponíveis no CKM não estarem preparados para serem usados de forma cíclica.

No entanto, é muito mais fácil criar ciclos com um tipo de "recomendação", por exemplo, especificar algo como A-> B> A, ou seja, um arquétipo (de algum tipo A), incluindo um *slot* que aceita um arquétipo genérico (de algum tipo B), incluindo um *slot* que pode aceitar. E esse seria o ciclo mais curto para resolver um problema. Claro que é possível ter ciclos mais longos.

Portanto, a questão é desvendar alguns questionamentos do tipo:

- Existe alguma recomendação para evitar ciclos? Por exemplo, poderia dizer que "você não pode anexar um arquétipo de mesmo ou maior nível (em relação a tipo de dados) para um *slot* ".

- Fazer referência é algo indispensado quando se modela dados clínicos? Em outras palavras, é realmente um caso de uso e não pode ser proibido. Com isso requer tomar mais cuidado;

- É possível em primeiro lugar detectar os ciclos, resolver as referências e, então criar um MR / GUI / ou outro artefato, de forma que não se torne exaustivo seguir os *links*?

Note mais uma possibilidade, seria, por exemplo, criar uma GUI com todas as informações disponíveis e um botão "Preencher este entalhe" no *slot*. Depois de clicar o botão, outra forma seria criada, que recolhe os dados para o arquétipo de referência. Neste caso, o algoritmo não atingiu um "*loop*", mas o usuário encontraria "novos botões" para seguir preenchendo os dados.

De acordo com essas informações, o fato de um filho de atributo precisar da inclusão de um outro arquétipo (através do uso de *slots*), fez com que a renderização do arquétipo proposto parasse, tornando o processo *parsing* instável, porque nesse caso precisaria ser gerada uma outra tela para a inclusão do *parse* desse arquétipo de referência. Sendo assim, somente se o primeiro filho renderizado tivesse definido o seu tipo. Isto é, se é um campo texto (*input-text*) ou outro campo qualquer, que permitisse a construção de um GUI, logo seria possível utilizar uma *label* (elemento de controle gráfico que exibe texto em um formulário) através dos termos e representar a renderização proposta em tela.

6 TRABALHOS FUTUROS

Durante a revisão da literatura foi possível encontrar trabalhos com a mesma proposta de gerar automaticamente uma *interface* gráfica através de arquétipos, levando em consideração a interoperabilidade semântica da informação, e também um trabalho no qual foi utilizado arquétipos openEHR para gerar automaticamente uma *interface* ao usuário, baseado em ADL.

Este projeto continuará com essa proposta de encontrar a melhor forma de se obter uma *interface* gráfica a partir da renderização de arquétipos, de forma que o sistema proposto fique estável e funcione por referência cíclica, ou seja, automática.

Isso me faz pensar que fazer algum algoritmo de preenchimento de *slot* de potencial altamente automatizado que tenta adivinhar todos os possíveis caminhos de dados permitidos pelas definições de *slots*, dessa forma quando começar um *cluster*, em teoria tudo se tornaria mais recursivo, levando a representação gráfica dos ciclos.

Essa forma de pensar é levada em consideração apenas para trabalhos futuros, justamente devido os atuais *slots* não possuírem semântica compatível, e não estarem com suas definições atualizadas. Portanto, por enquanto, é preciso evitar qualquer abordagem excessivamente mecanicista para gerar possíveis dados.

Para melhor renderização de arquétipos,o ponto de partida será o arquétipo juntamente com as descrições de restrições dispostas no AOM, sendo as regras sobre o que *slots* permitem, sejam definidas pela MR e elas funcionem exatamente como remédio para o problema. De fato espera-se que novas versões do ADL que virão supere todas essas expectativas.

Todos esses fatores devem ser levados em consideração para um futuro aperfeiçoamento das técnicas de renderização de arquétipos openEHR. Pois, eles compartilham a inovação chave do openEHR que é a adaptabilidade, e devido os arquétipos serem externos ao *software*, parte significante da aplicação pode ser gerada automaticamente a partir dos mesmos.

REFERÊNCIAS BIBLIOGRÁFICAS

ALMEIDA, V. P. Estratégias cognitivas para o aumento da qualidade do hiperdocumento para educação à distância. Dissertação (Mestrado) - Curso de Ciência da Computação, Centro de Ciências Exatas e Tecnologia, Universidade Federal de São Carlos, São Carlos, 2005..

ASTAH_COMMUNITY, 2014. Site Oficial Astah Comumunity. Disponível em < http://astah.net/> Acesso em 09/06/2014.

BEALE T; H. S. *Archetype Definition Language* 1.4.Rev. 1.4.1. The OpenEHR OPENEHR. Disponível em: <http://www.openehr.org >. Acesso em: 10 jan. 2014.

BEALE, T. *Archetype Object Model*. London: The OpenEHR Foundation, 2008.Disponível em:<http://www.openehr.org/svn/specification/TRUNK/publishing/architecture/am/aom1.4.pdf>.

BEALE, T. Archetypes: *Constraint-based Domain Models for Future-proof Information Systems*. Eleventh OOPSLA Workshop on Behavioral Semantics: Serving the Customer. Boston: 2002, pp. 16-32.

BITTENCOURT J, J. A. S. Editor de templates para um RES Semanticamente interoperável. Trabalho de Conclusão de Curso (Graduação) – Instituto Tecnólogico de Aeronáutica, São José dos Campos. 2009.

BRASIL, M. S.. Portaria nº 2.073 - Regulamenta o uso de padrões de interoperabilidade e informação em saúde para sistemas de informação em saúde no âmbito do Sistema Único de Saúde, nos níveis Municipal, Distrital, Estadual e Federal, e para os sistemas privados e do se. 2011.

BROOKSHEAR, J. G. Ciência da Computação - Uma Visão Abrangente. Porto alegre: Bookman, 2003.

BUCHENAU, M; SURI, J. F. " Experience prototyping". In: Simpósio sobre projeto de sistemas interativos, 2000. Proceedings on Designing Interative Systems: Processes, Practices and Techniques. Nova Iorque: ACM Press, 2000, p. 424-433.

CEE-IS. Comissão de Estudos Especiais em Informática e Saúde. Disponível em: <http://www.gt1-arquitetura.com.br/>. Acesso em 15 Jun 2014.

CYBIS, W. de A. Ergonomia de Interfaces Homem-Computador. Apostila para o Curso de Pós-Graduação em Engenharia de Produção, UFSC, 2000. Disponível em: http://www.labiutil.inf.ufsc.br/apostila.htm. Acesso em 07 Fev 2014.

CYBIS, W. de A. Engenharia de Usabilidade: uma abordagem ergonômica. 2003. Disponível em: < http://www.labiutil.inf.ufsc.br/hiperdocumento/conteudo.html >. Acesso em: 05Fev 2014.

DAMAS, L. SQL Structured Query Language. Rio de Janeiro: LCT, 2007.

DATE, C. J. Introdução a Sistemas de Bancos de Dados. Rio de Janeiro: Elsevier: 2003.

DE SOUZA, C. S. et al Projeto de Interfaces de Usuário: Perspectivas Cognitiva s e Semióticas. In: Congresso da Sociedade Brasileira de Computação– Jornada de Atualização em Informática. Rio de Janeiro. Anais, 1999.

DEITEL, H. M, Deitel, P. J. Java - Como Programar. Ed. Pearson, 2006.

DIAS, T. et al Estudo dos Conceitos Presentes em um Resumo de Alta Hospitalar para a Representação das Informações por meio de Arquétipos OpenEHR. I Workshop Ibero-Americano de Sistemas Interoperáveis em Saúde. Medicina (Ribeirão Preto) abril/2014; 47 (Supl. 1): 74-77. Disponível em<http://revista.fmrp.usp.br/2013/suplementos/revista_IASIS2014.pdf>. Acesso em: 10 Mai. 2014.

GAETE, R. A. C. Modelo de Interoperabilidade Semântica Aplicado ao Domínio da Saúde: Um Estudo de Caso na Vigilância Alimentar e Nutricional. 2012. Dissertação (Mestrado em Informática) – Programa de Pós Graduação em Informática, UnB, Brasília.

GONZÁLEZ, C. Visual Design of Interaction, Dialog, or Interface. SIGCHI Bulletin (ACM Press), New York, NY, v. 27, n. 1, p. 12-13, Jan. 1995.

GUBIANI et al Interoperabilidade Semântica do Prontuário Eletrônico do Paciente. Simpósio de Informática da Região Centro. Rio Grande do Sul: UFSM, 2003.

HARTSON, H. R. “Human-Computer Interaction: Interdisciplinary roots and trends”. In The Journal of System and Software. 1998.

HEUSER, C. A. Projeto de Banco de Dados. Porto Alegre: Sagra, 1998.

INFOESCOLA. Disponível em: <http://www.infoescola.com/engenharia-de-software/uml/>. Acesso em: 27 junho 2013.

ISO. ISO 9241-11: Ergonomic requirements for office work with visual display terminals (VDTs). Part 11 — Guidelines for specifying and measuring usability. Gènève. 1997.

ISO/TC251 13606-1:2008: Health informatics - Electronic health record communication - Part 1: Reference model, 2008.

JAVA. Disponível em: <http://www.java.com/pt_BR/download/whatis_java.jsp>. Acesso em: 11 junho 2014.

KIRAKOWSKI, J. SUMI: The de facto industry standardevaluation questionnaire for assessing quality of use of software by end users. Disponível em: <http://www.ucc.ie/hfrg/questionnaires/sumi/index.html>. Acesso em: 04 Março. 2014.

KONDO, M. N. S.. Mapeamento da base de conhecimento fundamentado em arquétipos: contribuição à informática em saúde. Tese (doutorado em Engenharia de Sistemas Eletrônicos) – Escola Politécnica da Universidade de São Paulo, USP, São Paulo. 2012.

KOPANITSA, G.; Tsvetkova, Z.; VESELI, H. Análise de Métricas de Avaliação de Usabilidade de Sistemas de Registro Eletrônico de Saúde. In: Projetos de Grande Escala em EHealth: Parceria em Modernização: Anais da Conferência EFMI Tópico Especial, Moscou, Rússia . 2012.

MACK, R.; NIELSEN, J. Usability Inspection Methods. New York, NY: John Wiley & Sons. 1994.

MARIANI, A. C. O Mundo dos Atores: uma perspectiva de introdução à programação orientada a objetos. Disponível em <http://www.inf.ufsc.br/poo/atores/sbie98/sbie98-atores.html> Acessado em 06/04/2010.

MEDEIROS, Ernani. Desenvolvendo software com UML 2.0 - definitivo - Publisher Pearson Education do Brasil - São Paulo - SP, 2004.

MORAN, T. “The Command Language Grammars: a represetantion for the user interface of interactive computer systems”. International Journal of Man-Machine Studies, 15, 3-50. 1981.

MYSQL, 2014. Site Oficial MySQL. Disponível em < www.mysql.com/>Acessado em 09/04/2014.

NETBEANS, 2014. Site Oficial NetBeans. Disponível em < https://netbeans.org/> Acessado em 09/04/2014.

NIELSEN, J., 1993. *Usability Engineering.* California: AP Profissional.

NIELSEN, J.; MOLICH, R. Heuristic evaluation of user interfaces. In: ACM CHI´90 CONFERENCE, 1990, Seattle. Proceedings… Seattle, 1990. p. 249-256.

NIELSEN, J. Top Ten Mistakes in Web Design. Alertbox, maio 1996. Disponível em: <http://www.useit.com/alertbox/9605.html>. Acesso em: 20 mar. 2014.

NORMAN, D. (1986) “Cognitive Engineering”. In D. Norman & S. Draper (eds.) User Centered System Design. Hillsdale, NJ. Lawrence Erlbaum. pp.31- 61.

OPENEHR. Disponível em: <http://www.openehr.org>. Acesso em: 10 fev 2014.

OPENEHR. Disponível em: < http://www.openehr.org/wiki/display/spec/Slots,+naming+and+merging>. Acesso em: 15 Maio 2014a.

OPENEHR. Disponível em: < https://github.com/openEHR/java-libs/tree/master/adl-parser>. Acesso em: 20 Maio 2014b.

OPENEHR. Disponível em: < http://www.openehr.org/ckm/>. Acesso em: 10 fev 2014c.

OPENEHR. Disponível em: < http://www.openehr.org/wiki/display/dev/Data+Validation>. Acesso em: 10 fev 2014d.

PADUA, C. I. P. S. E. Engenharia de Usabilidade. Universidade Federal de Minas Gerais. Agosto, 2012. Disponível em http://homepages.dcc.ufmg.br/~clarindo/arquivos/disciplinas/eu/material/referencias/apostila-usabilidade.pdf. Acessoem: 15 Mai. 2014

PREECE, J. et al Human Computer Interaction. Essex, England: Addison-Wesley, 1994. 775 p.

PREECE, J.; ROGERS, Y.; SHARP, H. Design de interação. Bookman, 2005.

PRESSMAN, Roger S. Engenharia de Software. São Paulo : Makron Books, 1995.

QAMAR, R. Semantic mapping of clinical model data to biomedical terminologies to facilitate interoperability. Thesus of PhD University of Manchester, School of Computer Science, Manchester, UK, 2008. Disponível em: <http//:www.openehr.org/publications/archetypes/RQamar-PhD-Thesis-Mar2008.pdf>. Acesso em 10 jan. 2014.

REBELO, I. Proposta de uma Ferramenta de Verificação dos Procedimentos de Interação em Sistemas de Realidade Virtual / IrlaBocianoski Rebelo. – Florianópolis: Departamento de Engenharia de Produção da UFSC, 2004.

REZENDE, D. A. Engenharia de software e sistemas de informação. Rio de Janeiro: Brasport, 2005.

RICARTE, I. L. M., 2001. Programação Orientada a Objetos: Uma Abordagem com Java. Disponível em < http://www.dca.fee.unicamp.br/cursos/PooJava/Aulas/poojava.pdf > Acessado em 29/03/2011.

ROCHA, H. V. D.; BARANAUSKAS, M. C. C. Design e avaliação de interfaces humano-computador. Campinas: NIED/Unicamp, 2003.

RUIZ, L. M.Editor gráfico de correspondencias de alto nivel para arquetipos de Historia Clínica Electrónica. 2010. 117f. Proyecto Fin de Carrera – Escuela Técnica Superior de Ingeniería Informática, Universidad Politécnica de Valencia,Valencia, septiembre 2010.

RUMBAUGH, James, Ivar Jacobson e Grady Booch. Unified Modeling Language Reference Manual. Addison-Wesley, Reading - MA, 1999.

SANTOS, M. R. Documentos técnicos descrevendo a linguagem Archetype Definition Language (ADL) da Fundação openEHR, suas principais assertivas e exemplos de códigos ADL em utilização na comunidade internacional. Brasília, 2013. Ministério da Saúde,Produto nº 02 – Contrato nº BR/CNT/1200952. Departamento de Informática do SUS.

SANTOS, M. R. Sistema de registro eletrônico de saúde baseado na norma ISO 13606: aplicações na Secretaria de Estado de Saúde de Minas Gerais. 2011. Tese (doutorado em Ciências da Informação) – Programa de Pós Graduação em Ciências da Informação, UFMG, Minas Gerais.

SANTOS, M. R. Documentos técnicos contendo descrição da abordagem de modelagem multinível openEHR e seus benefícios para o desenvolvimento de sistemas de Registro Eletrônico de Saúde. Ministério da Saúde. Produto nº 01 – Contrato nº BR/CNT/1200952.001. Agosto, 2012.

SANTOS, M. R..; BAX, M. P. Modelagem de um Repositório Central Baseado em Arquétipos para Sistemas de RES Federados. Anais do XII Congresso Brasileiro de Informática em Saúde (CBIS 2010). Minas Gerais: UFMG, 2010.

SCHULER, T. et al Towards automatically generating graphical user interfaces from openEHR archetypes. Stud Health Technol Inform. 2006;124:221-6. PubMed PMID: 17108529.

SIERRA, K.; BATES, B. Use a cabeça! Java. Rio de Janeiro: Alta Books, 2007.

SOMMERVILLE, Ian. Software Engineering, 4th ed. - Wokingham : Addison-Wesley, 1992.

VICENTE, C.*et al* Registro de Saúde Eletrônico Dinâmico Baseado em OpenEHR. I Workshop Ibero-Americano de Sistemas Interoperáveis em Saúde. Medicina (Ribeirão Preto) abril/2014; 47 (Supl. 1): 74-77. Disponível em<http://revista.fmrp.usp.br/2013/suplementos/revista_IASIS2014.pdf>. Acesso em: 10 Mai. 2014.

XMind, 2014. Site Oficial XMind. Disponível em < http://www.xmind.net/> Acesso em 10/04/2014.

ANEXOS

ANEXO1: SUMÁRIO DE ALTA HOSPITALAR — PARTE 1: MODELO DE INFORMAÇÃO

Modelo de informação

A Tabela 1 abaixo exibe os elementos que são partes do modelo de informação do sumário de alta hospitalar. O método para descrever o modelo é o seguinte:
• Coluna 1 – Item/Nível – descreve o nível do elemento no modelo de informação;
• Coluna 2 – Ocorrência – descreve o número de vezes que o elemento deve/pode aparecer, onde:

[0..] – indica que o elemento não é obrigatório;
[1..] – indica que o elemento deve estar presente pelo menos uma vez;
[..n] – indica que o elemento pode ocorrer várias vezes;
[..1] – indica que o elemento só deve aparecer uma vez.

Tabela 1 — Modelo de informação do sumário de alta hospitalar

Item/ Nível	Ocorrência	Modelo de Informação	Tipo de dados
1	**[1..1]**	**Identificação do paciente e caracterização da internação**	
2	[1..1]	Nome completo	*String*
2	[1..1]	Nome completo da mãe	*String*
2	[1..1]	Data de nascimento	Data Aceitar datas parciais: AAAA; MM/AAAA; DD/MM/AAAA.
2	[1..1]	Nº do cartão SUS	String
2	[1..1]	Número do prontuário	String
2	[0..1]	Número de controle	String
2	[1..1]	Identificação do estabelecimento de saúde (CNES)	Número CNES do estabelecimento de saúde
2	[1..1]	Caráter da Internação	Texto codificado: Eletiva; Urgência.
2	[1..1]	Data e Hora da Internação	DataHora

Tabela 1 (continuação)

Item/ Nível	Ocorrência	Modelo de Informação	Tipo de dados
2	[1..1]	Data e Hora da Saída da Internação	DataHora
2	[1..1]	Tipo de Internação	Texto codificado: Clínica; Cirúrgica; Pediátrica; Obstétrica; Psiquiátrica.
2	[1..1]	Regime da Internação	Texto codificado: Hospitalar; Hospital-dia; Domiciliar
1	**[1..1]**	**Motivo da admissão, diagnósticos relevantes e patologias associadas desenvolvidas na internação**	
2	[1..1]	Código CID 10 do Diagnóstico Principal	Texto codificado por Terminologia CID10.
2	[0..N]	Código CID 10 de outros diagnósticos relacionados ao principal	Texto codificado por Terminologia CID10.
1	**[0..N]**	**Classificação Internacional de Funcionalidade e Incapacidade em Saúde (CIF)**	
2	[1..N]	Código CIF – 2004	Texto codificado por terminologia externa CIF
1	**[0..1]**	**Lista de Problemas**	
2	[1..N]	Problema conforme terminologia externa	Texto codificado por Terminologia externa.
3	[1..1]	Código do Problema	Código conforme terminologia externa.
3	[1..1]	Status do Problema	Texto codificado: Ativo; Inativo.
3	[1..1]	Terminologia que descreve o problema	-

4	[1..1]	Identificador OID da terminologia que descreve o problema	OID
4	[0..1]	Nome da terminologia que descreve o problema	Texto livre
1	**[1..N]**	**Procedimentos diagnósticos realizados**	
2	[1..1]	Código do Procedimento diagnóstico realizado na internação	Texto codificado por Terminologia externa.
2	[1..1]	Procedimento diagnóstico realizado na internação	Texto livre
2	[1..1]	Terminologia que descreve o procedimento diagnóstico realizado na internação	-
3	[1..1]	Identificador OID da terminologia de procedimento diagnóstico realizado na internação	OID
3	[0..1]	Nome da terminologia que descreve o procedimento diagnóstico realizado na internação	Texto livre
1	**[1..N]**	**Procedimentos terapêuticos realizados**	
2	[1..1]	Código do procedimento terapêutico realizado na internação	Texto codificado por Terminologia externa.
2	[1..1]	Texto do procedimento terapêutico realizado na internação	Texto livre
2	[1..1]	Terminologia que descreve o procedimento terapêutico realizado na internação	-
3	[1..1]	Identificador OID da terminologia que descreve o procedimento terapêutico realizado na internação	OID
3	[0..1]	Nome da terminologia do procedimento terapêutico realizado na internação	Texto Livre
1	**[0..N]**	**Procedimentos cirúrgicos realizados**	
2	[1..1]	Código do procedimento cirúrgico realizado na internação	Texto codificado por Terminologia externa.
2	[1..1]	Texto do procedimento cirúrgico	Texto livre
2	[1..1]	Terminologia que descreve o procedimento cirúrgico realizado na internação	-
3	[1..1]	Identificador OID da terminologia que descreve o procedimento cirúrgico realizado na internação	OID
3	[0..1]	Nome da terminologia do	Texto livre

		procedimento cirúrgico realizado na internação	
1	**[1..1]**	**Resumo da Internação (Histórico da Internação)**	
2	[1..1]	Descrição dos principais eventos da internação	Texto livre
1	**[0..N]**	**Alergias e Reações Adversas na Internação**	
2	[1..1]	Categoria do agente causador da alergia / reação adversa	Texto codificado: Alimento; Animal; Ingrediente não ativo da medicação; Medicação; Outras substâncias ou produtos químicos; Outros; Produto ambiental; Substâncias para contrastes de exame de imagens.
2	[1..1]	Agente/Substância específica	Texto livre
2	[0..1]	Categoria da alergia / reação	Texto codificado: Alergia; Intolerância; Sem reação; Sensibilidade.
2	[0..1]	Intensidade da alergia / reação	Texto codificado: Fatal; Grave; Leve; Leve para Moderada; Moderada; Moderada para Grave.
2	[0..1]	Evolução da alergia / reação adversa	-
3	[0..1]	Data da instalação da reação adversa	DataHora. Aceitar parciais: Data e hora parcial; Data completa;

			Data parcial; Hora completa hh:mm:ss; Hora parcial; Hora parcial com minutos.
3	[0..1]	Duração da reação adversa	PT0S*
2	[0..1]	Exposição	-
3	[0..1]	Data da exposição	DataHora. Aceitar parciais: Data e hora parcial; Data completa; Data parcial; Hora completa hh:mm:ss; Hora parcial; Hora parcial com minutos.
3	[0..1]	Duração da exposição	PT0S*
1	**[1..1]**	**Prescrição e Orientação da Alta**	
2	[0..N]	Prescrição da alta (texto livre)	Texto livre
2	[0..1]	Lista de Medicamentos da Alta (estruturada)	Prescrição estruturada – ou a forma em texto livre ou a estruturada deve estar preenchida
3	[1..1]	Terminologia que descreve o medicamento	-
4	[1..1]	Identificador OID da terminologia de medicamentos	OID
4	[0..1]	Nome da terminologia de medicamentos	Texto livre
3	[1..N]	Linha de Prescrição	
4	[1..1]	Nome do medicamento (código)	Texto codificado por terminologia (informar qual)
4	[1..1]	Dose	Quantidade
4	[1..1]	Via de administração	Texto codificado por terminologia externa conforme Vocabulário Controlado de Formas

			Farmacêuticas, Vias de Administração e Embalagens de Medicamentos da ANVISA [21]
4	[1..1]	Frequência de uso da medicamento	Intervalo de tempo
4	[1..1]	Duração de uso da medicamento	Intervalo de tempo
4	[0..1]	Orientação sobre o uso do medicamento	Texto livre
1	**[0..1]**	**Instruções (educação ao paciente e familiar) Orientações; recomendações.**	**Texto livre**
1	**[0..1]**	**Informações da alta hospitalar**	
2	[1..1]	Condição do paciente	Texto codificado: curado; melhorado; inalterado.
2	[1..1]	Destino da alta	Texto codificado: Encaminhamento para outro profissional ou especialidade; alta com previsão de retorno; assistência domiciliar; hospital dia.
3	[0..1]	Local para acompanhamento pós-alta	-
4	[1..1]	Nome do estabelecimento de saúde / profissional para encaminhamento pós-alta	Texto livre
3	[1..1]	Profissional Responsável pela Internação	
4	[1..1]	Nome do médico responsável pela internação	Texto livre
4	[1..1]	CRM do médico responsável pela internação	Texto livre
4	[1..1]	UF do CRM do médico responsável pela internação	Texto codificado por terminologia IBGE

ABNT/CEE-78
1º PROJETO 78:000.01-021
FEV 2014

1	**[0..N]**	**Anexos**	
2	[1..1]	Descrição dos anexos	Texto livre

ANEXO2: CÓDIGO ADL DO ARQUÉTIPO OPENEHR-DEMOGRAPHIC-PERSON.PERSON-PATIENT.V1

```
archetype (adl_version=1.4)
	openEHR-DEMOGRAPHIC-PERSON.person-patient.v1
specialise
	openEHR-DEMOGRAPHIC-PERSON.person.v1
concept
	[at0000.1]	-- Dados do paciente
language
	original_language = <[ISO_639-1::pt-br]>
	translations = <
		["en"] = <
			language = <[ISO_639-1::en]>
			author = <
				["name"] = <"Sergio Miranda Freire">
				["organisation"] = <"Universidade do Estado do Rio de
Janeiro - UERJ">
				["email"] = <"sergio@lampada.uerj.br">
			>
		>
	>
description
	original_author = <
		["name"] = <"Sergio Miranda Freire & Rigoleta Dutra Mediano Dias">
		["organisation"] = <"Universidade do Estado do Rio de Janeiro - UERJ">
		["email"] = <"sergio@lampada.uerj.br">
```

["date"] = <"22/05/2009">

>

details = <

["en"] = <

language = <[ISO_639-1::en]>

purpose = <"Representation of a patient's demographic data.">

use = <"Used in demographic service to collect a patient's data.">

keywords = <"demographic service", "patient's data">

misuse = <"">

copyright = <"© 2011 openEHR Foundation">

>

["pt-br"] = <

language = <[ISO_639-1::pt-br]>

purpose = <"Representação dos dados demográficos de um paciente.">

use = <"Usado em serviço demográficos para coletar os dados de um paciente.">

keywords = <"serviço demográfico", "dados de um paciente">

misuse = <"">

copyright = <"© 2011 openEHR Foundation">

>

>

lifecycle_state = <"Authordraft">

other_contributors = <>

other_details = <

["references"] = <"ISO/TS 22220:2008(E) - Identification of Subject of Care - Technical Specification - International Organization for Standardization.">

>

```
definition
   PERSON[at0000.1] matches {  -- patient demographic data
      details matches {
         allow_archetype ITEM_TREE[at0001] occurrences matches {1..1} matches {
            include
               archetype_id/value matches {/(person_details)[a-zA-Z0-9_-]*\.v1/}
         }
      }
      identities matches {
         allow_archetype PARTY_IDENTITY[at0002.1] occurrences matches {1..1}
matches {
            include
               archetype_id/value matches {/(person_name)[a-zA-Z0-9_-]*\.v1/}
         }
      }
      contacts matches {
         CONTACT[at0003.1] occurrences matches {1..1} matches {  -- Contacts
            addresses matches {
               allow_archetype ADDRESS[at0030] occurrences matches {1..1} matches {
                  include
                     archetype_id/value matches {/(address)([a-zA-Z0-9_]+)*\.v1/}
                     archetype_id/value matches {/(electronic_communication)[a-zA-Z0-9_-
]*\.v1/}
               }
            }
         }
      }
      relationships matches {
```

```
                PARTY_RELATIONSHIP[at0004.1] matches {     -- personal
relationships
                    details matches {
                        ITEM_TREE [at0.40] matches {
                            items matches {
                                ELEMENT[at0040] matches {       --
type of relationship
                                    value matches {
                                        DV_TEXT matches {*}
                                        DV_CODED_TEXT
matches {
                                            defining_code
matches {[ac0000]}
                                        }
                                    }
                                }
                            }
                        }
                    }
                PARTY_RELATIONSHIP[at0.2] matches {         -- relationship
between a patient and a third-party payer
                    details matches {
                        ITEM_TREE[at0.20] matches {
                            items matches {
                                allow_archetype CLUSTER[at0.21]
        matches {
                                    include
                                    archetype_id/value matches
{/(person_identifier)[a-zA-Z0-9_-]*\.v1/}
```

```
                                        }
                                    }
                                }
                            }
                        }
                        PARTY_RELATIONSHIP[at0.3] matches {        -- relationship between a patient and a healthcare provider organisation/individual provider
                            details matches {
                                ITEM_TREE[at0.30] matches {
                                    items  matches {
                                        allow_archetype CLUSTER[at0.31] matches {
                                            include
                                            archetype_id/value matches {/(person_identifier)[a-zA-Z0-9_-]*\.v1/}
                                        }
                                    }
                                }
                            }
                        }
    }
}
ontology
    term_definitions = <
        ["pt-br"] = <
            items = <
                ["at0000.1"] = <
                    text = <"Dados do paciente">
```

```
                description = <"Dados do paciente.">
            >
            ["at0001"] = <
                text = <"Detalhes">
                description = <"Detalhes demográficos do paciente.">
            >
            ["at0002"] = <
                text = <"Nome">
                description = <"Conjunto de dados que especificam o
nome da pessoa.">
            >
            ["at0003"] = <
                text = <"Contatos">
                description = <"Contatos da pessoa.">
            >
            ["at0004"] = <
                text = <"Relacionamentos">
                description = <"Relacionamentos de uma pessoa,
especialmente laços familiares.">
            >
            ["at0002.1"] = <
                text = <"Nome">
                description = <"Conjunto de dados que especificam o
nome do paciente.">
            >
            ["at0003.1"] = <
                text = <"Contatos">
                description = <"Contatos da pessoa.">
```

>

["at0004.1"] = <

text = <"Relacionamentos">

description = <"Relacionamentos de um paciente, especialmente laços familiares.">

>

["at0030"] = <

text = <"Endereço">

description = <"Endereços vinculados a um único contato, ou seja, com o mesmo período de validade.">

>

["at0040"] = <

text = <"Grau de parentesco">

description = <"Define o grau de parentesco entre as pessoas envolvidas.">

>

["at0.2"] = <

text = <"Fonte pagadora">

description = <"Beneficiário: Relacionamento do paciente com uma Fonte Pagadora.">

>

["at0.3"] = <

text = <"Serviço de saúde/Profissional de saúde">

description = <"Relacionamento do paciente com uma organização prestadora de serviço de saúde/profissional de saúde.">

>

["at0.40"] = <

text = <"Personal relationships">

description = <"Personal relationship.">

>

["at0.20"] = <

text = <"Identificações do beneficiário">

description = <"Identificações do beneficiário junto à fonte pagadora.">

>

["at0.21"] = <

text = <"Identificação do beneficiário">

description = <"Documento de identificação do beneficiário junto à fonte pagadora.">

>

["at0.30"] = <

text = <"Identificações no prestador">

description = <"Identificações do paciente junto ao prestador.">

>

["at0.31"] = <

text = <"Identificação no prestador">

description = <"Documento de identificação do paciente junto ao prestador.">

>

>

>

["en"] = <

items = <

["at0000.1"] = <

text = <"Patient">

description = <"Patient demographic data.">

>

```
["at0001"] = <
        text = <"Demographic details">
        description = <"A patient's demographic details.">
>
["at0002"] = <
        text = <"Name">
        description = <"A person's name.">
>
["at0003"] = <
        text = <"Contacts">
        description = <"A person's contacts.">
>
["at0004"] = <
        text = <"Relationships">
        description = <"A person's relationships, especially family ties.">
>
["at0002.1"] = <
        text = <"Name">
        description = <"A patient's name.">
>
["at0003.1"] = <
        text = <"Contacts">
        description = <"A patient's contacts.">
>
["at0004.1"] = <
        text = <"Relationships">
```

description = <"A patient's relationships, especially family ties.">

>

["at0030"] = <

text = <"Addresses">

description = <"Addresses linked to a single contact, i.e. with the same time validity.">

>

["at0040"] = <

text = <"Relationship type">

description = <"Defines the type of relationship between related persons.">

>

["at0.2"] = <

text = <"Third party payer">

description = <"Relationship between the patient and a third-party payer.">

>

["at0.3"] = <

text = <"Healthcare provider/Health professional">

description = <"Patient: relationship between the patient and a healthcare provider organisation/health professional.">

>

["at0.40"] = <

text = <"Personal relationship">

description = <"Personal relationship.">

>

["at0.20"] = <

text = <"Patient identifiers">

```
                description = <"Identifiers of the patient at the third-
party payer.">
            >
            ["at0.21"] = <
                text = <"Healthcare consumer identifier.">
                description = <"An identifier of the patient at the
third-party payer.">
            >
            ["at0.30"] = <
                text = <"Patient identifiers">
                description = <"Patient identifiers at the related
healthcare provider.">
            >
            ["at0.31"] = <
                text = <"Patient identifier">
                description = <"A patient identifier at the related
healthcare provider.">
            >
        >
    >
>
constraint_definitions = <
    ["pt-br"] = <
        items = <
            ["ac0000"] = <
                text = <"Códigos para tipo de parentesco">
                description = <"códigos válidos para tipo de
parentesco.">
            >
```

```
                >
        >
        ["en"] = <
                items = <
                        ["ac0000"] = <
                                text = <"Codes for the type of relationship">
                                description = <"Valid codes for the type of
relationship.">
                        >
                >
        >
>
```