Pós-Graduação em Ciência da Computação

Andreza Leite de Alencar

Um Meta-modelo para Representação de Dados Biológicos Moleculares e Suporte ao Processo de Anotação de Variantes Genéticas

Recife

2018

Andreza Leite de Alencar

Um Meta-modelo para Representação de Dados Biológicos Moleculares e Suporte ao Processo de Anotação de Variantes Genéticas

Este trabalho foi apresentado à Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Doutor em Ciência da Computação.

Área de Concentração: Ciências Exatas e da Terra/Ciência da Computação

Orientador: Vinícius Cardoso Garcia
Co-Orientador: Vanilson André de Arruda Burégio

Recife
2018

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

A368m Alencar, Andreza Leite de
Um meta-modelo para representação de dados biológicos moleculares e suporte ao processo de anotação de variantes genéticas / Andreza Leite de Alencar. – 2018.
152 f.: il., fig., tab.

Orientador: Vinícius Cardoso Garcia.
Tese (Doutorado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2018.
Inclui referências e apêndices.

1. Engenharia de software. 2. Meta-modelo. I. Garcia, Vinícius Cardoso (orientador). II. Título.

005.1 CDD (23. ed.) UFPE- MEI 2018-139

Andreza Leite de Alencar

Um meta-modelo para representação de dados biológicos moleculares e suporte ao processo de anotação de variantes genéticas

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Doutora em Ciência da Computação.

Aprovado em: 06/09/2018.

_____

Orientador: Prof. Dr. Vinícius Cardoso Garcia

BANCA EXAMINADORA

_____

Profa. Dra. Ana Carolina Brandão Salgado
Centro de Informática /UFPE

_____

Prof. Dr. Robson do Nascimento Fidalgo
Centro de Informática / UFPE

_____

Prof. Dr. José Laurindo Campos dos Santos
Núcleo de Biogeo Informática /INPA

_____

Prof. Dr. Alexandre Magno Andrade Maciel
Escola Politécnica de Pernambuco/UPE

_____

Prof. Dr. Ricardo Argenton Ramos
Colegiado de Engenharia da computação/ UNIVASF

RESUMO

Na última década, surgiu uma nova tendência de abordagens que considera modelos não apenas como artefatos de documentação, mas também como artefatos centrais no campo da engenharia de software, permitindo a criação ou execução de sistemas de software complexos a partir desses modelos. Essas abordagens foram genericamente classificadas como *Model-Driven Engineering* (MDE). A análise de dados biológicos moleculares envolve a geração e interpretação de dados de genoma por sistemas de software complexos para apoiar decisões clínicas em diagnósticos. Esta análise pode ser decomposta em: (1) avaliação de qualidade dos dados brutos, (2) alinhamento de leitura a um genoma de referência, (3) identificação da variante, (4) anotação das variantes e visualização de dados. Por mais que existam diversas ferramentas para dar suporte a partes específicas deste processo, ele ainda enfrenta desafios. Um destes desafios está relacionado ao fato de não existirem padrões para publicação destes dados. Cada publicador escolhe qual conjunto de dados publicar e como publicá-lo. Como resultado, os serviços existentes, os formatos e os esquemas de dados podem variar significativamente. Diante deste cenário, identificou-se a demanda por pesquisas e soluções que possibilitem uma representação destes dados, auxiliando o processo de análise, especificamente na preparação de dados da etapa de anotação de variantes de genoma. Assim, esse trabalho busca responder a seguinte pergunta de pesquisa "*Quais os conceitos e características do domínio de dados biológicos moleculares que precisam ser identificados e mapeados para prover uma representação deste conjunto de dados e possibilitar a geração de ferramentas que possibilitem o gerenciamento de seus esquemas de dados?*". A abordagem de MDE surge como uma alternativa promissora neste cenário pois, com uso de recursos como meta-modelos e transformações de modelos, será possível contribuir com esta demanda. Assim, esse trabalho propõe uma solução que irá representar os esquemas de dados biológicos moleculares por meio de meta-modelos, permitindo o desenvolvimento de linguagens de modelagem e outros recursos que irão compor uma arquitetura de referência para dar suporte ao processo de anotação de variantes genéticas. A principais contribuições desse trabalho foram validadas por meio de suas implementações e avaliadas por meio de estudos baseados em opinião de especialistas e observação participante, que coletaram dados qualitativos e quantitativos sobre as contribuições. Entre as principais contribuições pode-se destacar: o meta-modelo GenDB; a linguagem de modelagem GenML; os algoritmos para identificação de esquemas e geração de esquemas de bases de dados orientadas a documento; e aplicação da abordagem de MDE para o desenvolvimento de soluções no contexto de engenharia de dados.

Palavras-chaves: Dados Biológicos Moleculares. Anotação de Variantes Genéticas. Meta-modelo. MDE. DSML

ABSTRACT

In the last decade, a new trend of approaches has emerged which considers models not only as documentation artifacts but also as central artifacts in the field of software engineering, allowing the creation or execution of complex software systems from these models. These approaches were generically classified as Model-Driven Engineering (MDE). The analysis of biomolecular data involves the generation and interpretation of genome data by complex software systems to support clinical decisions in diagnostics. This analysis can be broken down into: (1) raw data quality assessment, (2) reading alignment to a reference genome, (3) variant identification, (4) variant annotation and data visualization. Although there are several tools to support specific parts of this process, it still faces challenges. One of these challenges is related to the fact that there are no standards for publication of this data. Each publisher chooses which dataset to publish and how to publish it. As a result, existing services, formats, and data schemas can vary significantly. In this scenario, we identified the demand for research and solutions that allow a representation of these data, assisting the analysis process, specifically in the preparation of data from the annotation step. Thus, this theses seeks to answer the following research question *"Which are the concepts and characteristics of the biomolecular data domain that need to be identified and mapped to provide a representation of this data set and enable the generation of tools that enable the management of your data schemas?"*. The MDE approach emerges as a promising alternative in this scenario because, with the use of resources such as meta-models and model transformations, it will be possible to contribute to this demand. Thus, this work proposes a solution that will represent the biomolecular data schemas by means of meta-models, allowing the development of modeling languages and other resources that will compose a reference architecture to support the genetic variants annotation process. The main contributions of this work were validated through their implementations and evaluated through expert opinion and participant observation studies, which collected qualitative and quantitative data on contributions. Among the main contributions, can be highlighted: the GenDB meta-model; the GenML modeling language; algorithms for schema identification and generation of document-oriented database schemas; and application of the MDE approach to the development of solutions in the context of data engineering.

Key-words: Biomolecular Data. Genome Variant Annotation. Meta-model. MDE. DSML.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

ACID	Atomicidade, Consistência, Isolamento, Durabilidade
API	*Application Programming Interface*
BASE	*Basically Available, Soft-State, Eventually Consistent*
BNF	*Backus-Naur Form*
CAP	*Consistency, Availability, Partition Tolerance*
CASE	*Computer-Aided Software Engineering*
CO	Consultas ao Observador
DI	Diagramas Incompletos
DNV	Diagrama Não Validado
DSL	*Domain-Specific Language*
DSML	*Domain-Specific Modeling Language*
DTD	*Document Type Definition*
EA	Erros de Aplicação
EC	Erros de Compreensão
EF	Erros da Ferramenta
EMF	*Eclipse Modeling Framework*
ER	Entidade-Relacionamento
GQM	*Goal, Question, Metric*
JSON	*JavaScript Object Notation*
M2T	Modelo para Texto
M2M	Modelo para Modelo
MBE	*Model Based Engineering*
MBT	*Model Based Test*
MDA	Model Driven Architecture
MDD	*Model Driven Development*

MDE	*Model Driven Engineering*
MDSD	*Model Driven Software Development*
MIDST	*Model Independent Schema and Data Translation*
MIDSTRT	*Model Independent Schema and Data Translation - Run Time*
MPS	*Meta Programming System*
NGS	*Next Generation Sequencing*
NoSQL	*Not Only SQL*
ODM	*Object Document Mapper*
OEM	*Object Exchange Model*
OMG	*Object Management Group*
PU	Problemas de Usabilidade
QIP	*Quality Improvement Paradigm*
QS	Qualidade Semântica
SGBD	Sistema de Gerenciamento de Banco de Dados
SOS	*Save our Systems*
T2M	Texto para Modelo
TM	Tempo de Modelagem
UML	*Unified Modeling Language*
W3C	*World Wide Web Consortium*
URL	*Uniform Resource Locator*
VCF	*Variant Call Format*
XML	*eXtensible Markup Language*
XSD	*XML Schema Definition*

SUMÁRIO

1 INTRODUÇÃO

> *"To apply an experimental test would be to show ignorance of the difference between human nature and divine."*
>
> —Plato, *Timaeus and Critias*

Muitas técnicas e linguagens de modelagem foram propostas para suporte ao projeto e desenvolvimento de sistemas de software complexos. Muitas dessas linguagens foram definidas no contexto de abordagens metodológicas como estruturada, orientada a objetos ou processo/metodologia unificada, fundamentalmente com o objetivo de facilitar e compartilhar uma visão comum coerente do sistema sob investigação e, consequentemente, facilitando a comunicação entre os interessados na solução (BOOCH, 1994; WEAVER, 1993; YOURDON, 1989; KROGSTIE, 2012).

No entanto, durante a última década, surgiu uma nova tendência de abordagens que considera modelos não apenas como artefatos de documentação, mas também como artefatos centrais no campo da engenharia de software, permitindo a criação ou execução de sistemas de software a partir desses modelos. Essas abordagens foram genericamente classificadas como *Model-Driven Engineering* (MDE) (BRAMBILLA; CABOT; WIMMER, 2012).

O processo de análise de dados biológicos moleculares (para realização de diagnóstico molecular) consiste da geração (coleta) e interpretação de dados de genoma para apoiar as decisões clínicas. Isto pode impactar diretamente em diagnósticos de pacientes com câncer, por exemplo (CHIN; ANDERSEN; FUTREAL, 2011). Essa análise de dados de genoma humano pode ser decomposta em etapas distintas: (1) avaliação de qualidade dos dados brutos, (2) alinhamento de leitura a um genoma de referência, (3) identificação da variante, (4) anotação das variantes e visualização de dados(OIKAWA et al., 2004). Por mais que existam diversas ferramentas para dar suporte a partes específicas deste processo, este ainda enfrenta desafios (PABINGER et al., 2014).

Os principais desafios em lidar com estes dados, em aplicativos de saúde e genômicos, estão relacionados a: privacidade e segurança de dados, fluxo e armazenamento de dados, falta de treinamento em ciência de dados e, descoberta e acesso a dados biomédicos (IDC, 2014; MASSEROLI; CANAKOGLU; CERI, 2016; LIU; LIU; YANG, 2010; LIU et al., 2009; BALKO et al., 2004). Ainda existem demandas por soluções que permitam aos cientistas superar os problemas discutidos abaixo (MASSEROLI; CANAKOGLU; CERI, 2016; ALENCAR et al., 2016b; ALENCAR et al., 2016c; ALENCAR et al., 2016a):

- **Fontes de dados Sobrepostas e Descentralizadas**: Catálogos on-line de dados clínicos estão disponíveis na Web, mas não são unificados e são extremamente

sobrepostos. ClinVar[1] e OMIN[2] são exemplos de repositórios online com dados sobrepostos. Os consumidores de tais dados (por exemplo, clínicas e laboratórios de genética) têm dificuldades em criar novos aplicativos consistentes sobre eles. Isso acontece porque eles precisam de um esforço extra para analisar, entender e lidar com as informações extraídas dessas várias fontes.

- **Falta de Padrões**: além das fontes de dados sobrepostas e descentralizadas, não existem padrões para publicação destes dados. Cada publicador escolhe qual conjunto de dados e como publicá-lo. Muitas vezes, também não há acordo entre publicadores, laboratórios e clínicas. Como resultado, os serviços existentes que consomem dados clínicos, os formatos, esquemas e os tipos de dados podem variar significativamente.

- **Usabilidade**: é necessário construir sistemas internacionalizados com interfaces de usuário amigáveis para anotação de genoma e gerenciamento de esquemas de dados. Por este motivo, os usuários precisam trabalhar com arquivos de texto e planilhas complexas, tornando a análise de dados de genoma uma tarefa custosa.

- **Custo**: A falta de sistemas apropriados com interfaces de usuário amigáveis aumenta o custo do trabalho porque faz com que biólogos e geneticistas gastem muito tempo de trabalho em suas atividades para emissão de diagnósticos.

Diante deste cenário, identificou-se a demanda por pesquisas e planejamento de soluções complexas que possibilitem uma representação destes dados de genoma auxiliando o processo de análise de dados biológicos moleculares, especificamente na preparação de dados da etapa de anotação de variantes de genoma. Assim, este trabalho busca responder a seguinte pergunta de pesquisa "*Quais os conceitos e características do domínio de dados biológicos moleculares que precisam ser identificados e mapeados para prover uma representação deste conjunto de dados e possibilitar a geração de ferramentas que possibilitem o gerenciamento de seus esquemas de dados?*".

Neste cenário, a Engenharia Baseada em Modelos (MDE) (BRAMBILLA; CABOT; WIMMER, 2012) surge como uma alternativa promissora. Com o uso da abordagem de MDE (BRAMBILLA; CABOT; WIMMER, 2012; CHILLÓN et al., 2017; SIEGEL, 2014; FRANKEL, 2002; VOELTER et al., 2013; GREENFIELD; SHORT, 2003; MELLOR; BALCER, 2002; ATKINSON; KüHNE, 2003; STAHL; VOELTER; CZARNECKI, 2006; SELIC, 2008; KELLY; TOLVANEN, 2008; UTTING; LEGEARD, 2007; CHILLÓN et al., 2017), especialmente meta-modelos e transformações de modelos, será possível contribuir com esta demanda. A meta-modelagem é útil para construir representações em alto nível de abstração, permitindo que a informação seja uniformemente representada, e as transformações de modelo ajudam no desenvolvimento de ferramentas para o gerenciamento de esquemas. Ainda, com o uso de MDE,

[1] https://www.ncbi.nlm.nih.gov/clinvar/
[2] https://omim.org

parte da complexidade dos softwares a serem desenvolvidos fica concentrada dentro dos geradores de código, auxiliando os desenvolvedores e bioinformatas[3] na construção de ferramentas. Com isso, pode-se reduzir a incidência de erros e aumentar a produtividade, reuso, qualidade, interoperabilidade e manutenibilidade das ferramentas produzidas.

Neste contexto, o uso de técnicas de MDE pode efetivamente auxiliar no desenvolvimento de ferramentas que possam dar suporte ao processo de análise de dados biológicos moleculares (especificamente na preparação de dados da etapa de anotação de variantes de genoma). Assim, este trabalho propõe uma solução, baseada em MDE, que irá representar os esquemas de dados biológicos moleculares por meio de meta-modelos, permitindo o desenvolvimento de linguagens de modelagem e outros recursos que irão compor uma arquitetura de referência para o domínio. Entre estes recursos, uma abordagem para identificação de esquemas versionados será proposta. Será proposta também uma Linguagem de Modelagem Específica de Domínio (em inglês, *Domain Specific Modeling Language* [DSML]) para visualização de esquemas, aplicando uma abordagem generativa para gerar os diagramas (modelos) dos esquemas identificados. Estes modelos serão manipulados diretamente por transformações de modelo para obter a diagramação de esquemas e, posteriormente, a geração de esquemas de bacos de dados orientados a documento.

1.1 CARACTERIZAÇÃO DO PROBLEMA

Um laboratório de sequenciamento de genoma humano possui um ambiente caracterizado pela produção e uso de um volume de dados massivo oriundo de diversas fontes (IDC, 2014). De uma maneira geral, a análise de dados é um desafio enfrentado em genética clínica (IDC, 2014; MASSEROLI; CANAKOGLU; CERI, 2016; LIU; LIU; YANG, 2010; LIU et al., 2009; BALKO et al., 2004).

Na análise de variantes para aplicações de diagnóstico molecular, uma tarefa central é relacionar informações biológicas com dados clínicos de tal forma que permita aos especialistas determinarem o impacto potencial de uma determinada variante estar associada a uma doença. Esta tarefa é chamada formalmente de "anotação".

Para realizar anotações, é exigido a montagem flexível de conjuntos de dados sob medida e continuamente verificados, sem desperdiçar o tempo de biólogos e geneticistas em consultas individuais aos vários bancos de dados disponíveis on-line. Esta tarefa se torna custosa devido ao ciclo de consulta, análise, limpeza e visualização desses dados em planilhas complexas. Ainda, as bases que são consultadas nesta etapa são disponibilizadas em variados formatos que precisam ser tratados não só em nível da estrutura (esquema) mas também em relação ao modelo de dados nos quais estas fontes são publicadas, pois não existem padrões ou normas claras comuns para representação e consumo destes dados.

[3] Bioinformata é o profissional que exerce funções e atividades ligadas a bioinformática. Bioinformática é um campo interdisciplinar que corresponde à aplicação das técnicas da informática, no sentido de análise da informação, nas áreas de estudo da biologia.

Como exemplo de fontes públicas e privadas para dados de genoma pode-se citar ClinVar, OMIM, 1000 Genomes, RefGene, LOVD, ExAC65000, entre outros, além dos dados produzidos pelo próprio laboratório que irá realizar o diagnóstico.

Embora tenham havido alguns esforços na definição de ferramentas e soluções para melhorar o fluxo de trabalho da análise de dados de genoma (por exemplo, as propostas por (MASSEROLI; CANAKOGLU; CERI, 2016; PABINGER et al., 2014; LIU; LIU; YANG, 2010; LIU et al., 2009; BALKO et al., 2004; OIKAWA et al., 2004) que serão discutidas no Capítulo 3), a anotação ainda é considerada a etapa mais laboriosa, e que exige mais tempo. Com o uso de planilhas e bancos de dados, é possível realizar consultas, mas que dependem do time de bioinformática para preparar e entregar estes dados aos analistas. Diante deste cenário, esta pesquisa identificou a necessidade de definir uma representação que possibilite a construção de ferramentas que dêem suporte ao processo de análise de dados biológicos moleculares, especificamente na preparação dos dados da etapa de anotação de variantes de genoma.

A maioria destas bases são disponibilizadas em formatos de dados semi-estruturados (por exemplo, *eXtensible Markup Language*[XML]) onde esquemas não explicitamente definidos é um recurso intrínseco, pois fornece a flexibilidade necessária quando a estrutura de dados varia. No entanto, a equipe de bioinformática precisa entender a estrutura do esquema implícito nestas bases quando está desenvolvendo as ferramentas para suporte a anotação de variantes genéticas. Neste contexto, os bioinformatas têm que dedicar um esforço considerável para verificar se os dados gerenciados nos programas estão de acordo com as estruturas das fontes. Quando sistemas de banco de dados (por exemplo, sistemas relacionais) exigem a definição de um esquema que especifique a estrutura dos dados armazenados, uma verificação assegura que apenas os dados que se ajustam ao esquema podem ser manipulados no código do aplicativo e os erros cometidos pelos desenvolvedores são identificados com esta verificação. Por outro lado, os bancos de dados que não definem seus esquemas de maneira explícita exigem que os desenvolvedores garantam o acesso correto aos dados. Esta é uma tarefa propensa a erros, mais ainda quando entidades de mesmo tipo (que representam um mesmo conceito) podem apresentar variações na estrutura dos seus dados.

Isso demanda a definição de estratégias de engenharia reversa para identificar os esquemas (isto é, as diferentes estruturas em que os dados são disponibilizados) e construir ferramentas que usem os esquemas identificados para oferecer recursos que auxiliem os bioinformatas. Estas ferramentas trarão os benefícios da identificação de esquemas sem perder a flexibilidade dos dados semi-estruturados. A identificação dos vários esquemas de uma mesma entidade é um desafio que se torna maior pela presença de várias fontes de dados que publicam múltiplos conjuntos de dados periodicamente.

Assim, esse trabalho irá propor soluções que farão uso da abordagem de MDE, especialmente meta-modelos e transformações de modelos, para possibilitar a representação

dos dados de genoma e a implementação de ferramentas de apoio aos bioinformatas na preparação dos dados para anotação de variantes genéticas.

1.2 OBJETIVOS

O principal objetivo desse trabalho é especificar um meta-modelo para representação do conjunto de dados envolvido no processo de análise de dados biológicos moleculares (especificamente na etapa de anotação de variantes de genoma). Este meta-modelo deve possibilitar a implementação de ferramentas de apoio aos bioinformatas na preparação dos dados para anotação de variantes genéticas, auxiliando na identificação e conhecimento dos esquemas dos dados. A seguir, serão detalhados os objetivos específicos:

1. **OE1**: Analisar o domínio de dados biológicos moleculares. Esta análise deve identificar as principais características para modelagem, projeto e implementação do domínio. Entre estas características, devem ser identificados os conceitos necessários para representação das bases de dados biológicos moleculares;

2. **OE2**: Especificar um meta-modelo que explique/represente o conjunto de dados biológicos moleculares. Este meta-modelo deve definir quais são os conceitos do domínio e quais são as correspondências e relacionamentos entre estes conceitos.

3. **OE3**: Especificar uma linguagem de modelagem específica de domínio. Especificar uma DSML com notação visual para dados biológicos moleculares. Especificar diagramas que possam representar visualmente os diferentes tipos de esquemas identificados para bancos de dados biológicos moleculares.

4. **OE4**: Especificar uma abordagem de identificação de esquema. Especificar uma algoritmo de engenharia reversa baseada em modelos para identificar os esquemas implícitos em bancos de dados semi-estruturados XML. A abordagem deve levar em conta os conceitos presentes nas bases de dados biológicos moleculares.

5. **OE5**: Especificar uma abordagem para criação de bases de dados. Especificar um algoritmo de transformação de modelos para geração de esquemas de bases de dados orientadas a documento, a partir dos modelos criados com a DSML.

6. **OE6**: Implementar e Avaliar as soluções apresentadas nesse trabalho. Implementar as ferramentas necessárias para validação das soluções apresentadas nesse trabalho. Verificar a qualidade das soluções e analisar empiricamente os resultados obtidos.

1.3 CONTRIBUIÇÕES ESPERADAS

A principal contribuição esperada nesse trabalho é a definição de um meta-modelo capaz de representar os esquemas de bases de dados biológicos moleculares. Este meta-modelo também deve permitir a criação de ferramentas para apoio ao gerenciamento de esquemas dos dados envolvidos no processo de anotação de variantes de genoma. Entre estas ferramentas pode-se destacar uma DSML para visualização dos esquemas versionados das bases de dados biológicos moleculares, considerando o versionamento das entidades e das fontes de dados. Esse trabalho também irá apresentar uma abordagem de engenharia reversa baseada na transformação de modelos para identificar os esquemas das bases. A transformação de modelos também deve ser aplicada para definir algoritmos de geração de esquemas para bases de dados orientadas a documento.

Assim, as contribuições esperadas nesse trabalho podem ser resumidas em:

- Uma análise para identificação dos conceitos e tipos de esquema presentes nos bancos de dados biológicos moleculares, relacionada ao objetivo OE1;
- Um meta-modelo para representação de bases de dados biológicos moleculares, relacionado ao objetivo OE2;
- Uma linguagem de modelagem para visualização e diagramação dos esquemas das bases de dados biológicos moleculares, reacionada ao objetivo OE3;
- Um algoritmo para identificação de esquemas em bases de dados biológicos moleculares, relacionado ao objetivo OE4;
- Um algoritmo para geração de esquemas de bases de dados orientadas a documento, relacionado ao objetivo OE5; e
- A aplicação da abordagem de MDE para o desenvolvimento de soluções no contexto de engenharia de dados; relacionada ao objetivo OE6;

Escopo Negativo: É importante mencionar que esse trabalho não teve o objetivo de contribuir com o estado da arte de integração de dados e nem concentrou esforços para o tratamento de grandes volumes de dados no contexto de Big Data. Assim, o desempenho da manipulação dos dados de genoma e a integração destes dados não foram considerados como fatores norteadores para a definição das soluções propostas nesse trabalho.

1.4 METODOLOGIA DE PESQUISA

As atividades desse trabalho foram definidas com base na metodologia *Quality Improvement Paradigm* (QIP) (BASILI; GREEN, 1994) que fornece uma base para a aprendizagem e melhoria da qualidade da pesquisa. Em resumo, este é um processo iterativo no qual

o conhecimento produzido ao longo do processo por meio da construção e avaliação de novos artefatos serve como *feedback* para um melhor planejamento e implementação da solução final.

A Figura 1 mostra o ciclo tradicional do QIP, que é composto por seis etapas principais. A seguir serão apresentados os detalhes e atividades definidas para cada uma destas etapas.

Figura 1 – Ciclo do *Quality Improvement Paradigm* (QIP)

1. Caracterizar o ambiente do projeto - Esta etapa requer uma compreensão dos diversos fatores que podem influenciar o desenvolvimento do projeto. Nessa etapa, os principais conceitos, tecnologias, teorias e práticas aplicáveis ao domínio desta pesquisa foram levantados e estudados. A principal atividade realizada nessa etapa foi o entendimento e caracterização do problema, discutido na Seção 1.1. Nesta etapa também foi realizado o levantamento bibliográfico para fundamentação teórica e para conhecimento do estado da arte e identificação dos trabalhos relacionados com o problema investigado nesta pesquisa, que serão discutidos nos Capítulos 2 e 3, respectivamente.

2. Definir objetivos quantificáveis - Após aquisição do conhecimento necessário para mitigar o problema abordado na pesquisa, os principais objetivos do projeto devem ser determinados. A Seção 1.2 apresentou os principais objetivos definidos para esta pesquisa.

3. Escolher o modelo de processo - A metodologia QIP não é completa por si só pois esta não apresenta diretrizes orientadoras para a execução do projeto. A metodologia

QIP indica que o pesquisador escolha outro processo com seus métodos e ferramentas para dar suporte a execução do projeto. Assim, essa etapa tem o objetivo de escolher o modelo de processo que dará suporte à execução do projeto. Com base nos resultados das fases anteriores, os elementos-chave para compor a abordagem metodológica devem ser escolhidos. A principal atividade realizada nessa etapa foi a definição do processo e técnicas que serão utilizadas para execução do projeto. Esta atividade resultou na definição de um processo para engenharia de domínio baseada em modelos. Este processo será detalhado na Seção 1.4.1.

4. Executar - Executar o processo e construir os produtos necessários para composição da solução proposta. Coletar, validar e analisar os resultados para fornecer *feedback* em tempo real para ações corretivas.

5. Analisar resultados - Essa etapa irá analisar empiricamente os resultados obtidos na etapa anterior avaliando as práticas atuais para determinar problemas, registrar descobertas e fazer recomendações para futuras melhorias no projeto.

6. Empacotar a experiência - Os resultados da pesquisa devem ser empacotados em forma de processos refinados ou outras formas de conhecimento estruturado obtido a partir da pesquisa. Estes devem ser armazenados em uma base de experiência para que possam ser consultados e reutilizados em projetos futuros. Nesta etapa, serão que apresentam os resultados da pesquisa. Nessa pesquisa, esta etapa se caracteriza pela publicação das ferramentas, artigos e documento da tese.

Seguindo as atividades definidas, inicialmente foi identificado o problema e sua motivação para este trabalho. As ideias que serviram para estabelecer o propósito e execução desta tese vieram do conhecimento adquirido sobre Engenharia de Domínio, técnicas de MDE e a visão clara da necessidade ou conveniência de esquemas formalmente representados para desenvolver DSML e ferramentas *Computer-Aided Software Engineering* (CASE) para visualização de esquemas e suporte à análise de dados biológicos moleculares.

1.4.1 Processo para Engenharia de Domínio Baseada em Modelos

Com o intuito de analisar, projetar e implementar o domínio sob investigação nesse trabalho, foi necessário estudar diversas abordagens com propostas para engenharia de domínio, Reuso, *Model-Driven Engineering*(MDE) e *Model-Driven Development*(MDD) (KROGSTIE, 2012; BRAMBILLA; CABOT; WIMMER, 2012; VOELTER et al., 2013; LUCREDIO et al., 2008). Alguns trabalhos acadêmicos (LUCREDIO et al., 2008; KROGSTIE, 2012) apresentam abordagens sistemáticas com processos para análise de domínio com técnicas de MDD e MDE mas as considerações a respeito da qualidade dos artefatos gerados não são integradas aos processos e não apresentam meios de operacionalização para uma verificação de qualidade. Em (LUCREDIO et al., 2008) Lucredio *et. al* apresenta uma abordagem de

analise de domínio para reuso em projetos MDSE (*Model-driven Software Engineering*). O objetivo principal desse trabalho se concentrou em identificar as principais partes do domínio que podem ser implementadas usando técnicas de MDD para aumentar o nível de reuso do sistema. Com isso, identificou-se a necessidade de serem integradas a este adaptações das diretrizes de qualidade apresentadas em outros trabalhos relevantes, como o proposto por Krogstie (KROGSTIE, 2012), por exemplo. O resultado desta integração foi um processo para engenharia de domínio baseada em modelos que evidencia a importância da consideração dos critérios de qualidade durante a sua execução.

A Figura 2 apresenta o fluxo do processo utilizado para a execução da engenharia do domínio com as principais entradas (informações sobre o domínio e diretrizes de qualidade) e saídas (evidências e artefatos produzidos) das etapas de análise, projeto, implementação e avaliação. Nesta, observa-se que as fases de Análise, Projeto e Implementação podem ter múltiplas iterações até alcançarem níveis satisfatórios de detalhamento (especificação) e qualidade, além de um número suficiente de evidências para atendimento dos objetivos do projeto. A seguir, serão discutidas as principais características de cada fase do processo.

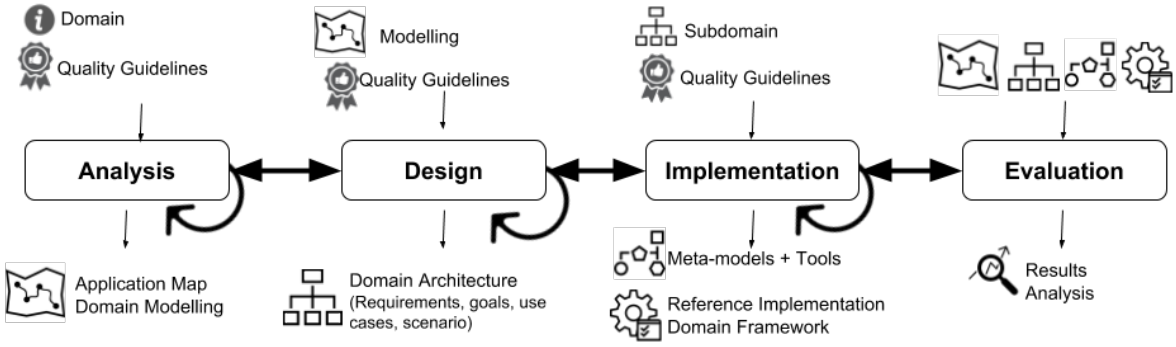

Figura 2 – Fluxo do processo

- **Análise - Objetivo:** a fase de análise do domínio tem como objetivo definir o escopo do domínio. Os objetivos desta fase podem ser classificados em Objetivos de Concepção e Objetivos de Validação que referem-se às entregas desejadas e ao nível de concordância sobre aos artefatos entre as partes interessadas, respectivamente. **Entrada:** esta fase recebe como entrada as informações disponíveis sobre o domínio e seus sistemas como o conhecimento de especialistas e *stakeholders*. **Saída:** a fase de análise tem como principal produto a documentação do domínio com: o mapa de aplicações do domínio, com suas respectivas *features* e a modelagem do domínio. **Qualidade:** para esta fase o conhecimento de especialistas e *Stakeholders* deve ser considerado aumentando assim os níveis de Completude, Validade e Integridade do projeto. Estes critérios estão relacionados ao Indicador de Qualidade Semântica do Projeto. As definições conceituais destes indicadores estão disponíveis na Tabela 1.

Atividades: as atividades e execução desta etapa serão apresentadas no Capítulo 4.

- **Projeto** - **Objetivo:** a fase de projeto do domínio tem como principal objetivo definir a arquitetura do domínio e seus componentes. Por este motivo, este pode ser descrito no nível de desenvolvimento e implementação dos artefatos. **Entrada:** esta fase recebe como entrada os artefatos produzidos na fase de análise. **Saída:** a fase de projeto tem como principal saída a definição da arquitetura do domínio com principais requisitos e detalhes sobre módulos, interfaces e artefatos existentes (ex. *Domain-Specific Language*[DSL], transformações, geradores de código). **Qualidade:** o conhecimento dos especialistas também deve ser considerado para aumentar os níveis de Simplicidade, Compreensibilidade, Completude, Validade e Integridade dos elementos identificados. Estes critérios estão relacionadas ao Indicador de Qualidade Empírica, Semântica, e Pragmática do projeto. As definições conceituais destes indicadores estão disponíveis na Tabela 1. **Atividades:** as atividades e execução desta a etapa serão apresentadas juntamente com a etapa de Análise no Capítulo 4.

- **Implementação** - **Objetivo:** o objetivo desta fase é implementar o domínio, ou seja, concretizar em forma de artefatos de implementação as informações coletadas sobre o domínio e planejadas durante a análise e projeto. **Entrada:** esta fase recebe como entradas os produtos das fases anteriores. **Saída:** como principal saída desta fase tem-se o chamado *framework* do domínio, que consiste do conjunto de artefatos de MDD (meta-modelos, DSLs, transformações e geradores de código) com a implementação de referência destes artefatos. **Qualidade:** os critérios considerados nesta fase são Implementabilidade, Flexibilidade, Reutilização e Minimalização. Estes critérios estão relacionados ao Indicador de Qualidade Organizacional e Qualidade Semântica. As definições conceituais destes indicadores estão disponíveis na Tabela 1. **Atividades:** as atividades e execução desta a etapa serão apresentadas no Capítulo 5.

- **Avaliação** - **Objetivo:** a fase de avaliação consiste num processo específico para verificação da qualidade dos artefatos gerados. **Entrada:** esta fase recebe como entrada todos os artefatos que serão avaliados. **Saída:** esta fase terá como saída os valores para cada um dos indicadores de qualidade analisados. **Qualidade:** esta fase irá considerar todos os indicadores de qualidade (Qualidade Empírica, Qualidade Sintática, Qualidade Semântica, Qualidade Pragmática e Qualidade Organizacional). As definições conceituais destes indicadores e os critérios que compõem cada um destes estão disponíveis na Tabela 1. **Atividades:** as definições para verificação da qualidade de meta-modelos serão apresentadas na Seção 1.4.1.1. As atividades e execução desta a etapa serão apresentadas no Capítulo 6.

Tabela 1 – Definição conceitual dos indicadores de qualidade. Adaptada de (KROGSTIE, 2013).

Indicadores	Definições	Critérios
Qualidade Empírica (KROGSTIE, 2013)	Lida com a compreensão e frequências de erros previsíveis quando um meta-modelo é lido ou escrito por diferentes atores. Define como o meta-modelo se apresenta em termos de ergonomia cognitiva, ou seja, é definido como a facilidade com que os conceitos e estruturas do meta-modelo de dados podem ser compreendidos, abrangendo questões de compreensão como layout para gráficos e índices de legibilidade para texto.	Compreensibilidade e Simplicidade
Qualidade Sintática (KROGSTIE, 2013)	É uma correspondência entre o meta-modelo e a extensão da linguagem (conjunto de todas as declarações possíveis de acordo com as regras da linguagem de modelagem usada). Tem o objetivo de corretude sintática, ou seja, que todas as declarações do meta-modelo estão de acordo com a sintaxe e o vocabulário da linguagem de modelagem. Em resumo, é a conformidade entre o meta-modelo e a sintaxe da linguagem de modelagem.	Corretude e Integridade
Qualidade Semântica (KROGSTIE, 2013)	Define como o meta-modelo reflete o conhecimento explícito (das partes interessadas) relevante para o domínio. É uma correspondência entre um meta-modelo e seu domínio (conjunto de todas as declarações que podem ser estabelecidas sobre a situação). Em resumo, verifica se todos os elementos do domínio são acomodados pelo meta-modelo.	Completude, Validade, Integridade, Integração, Reuso e Minimalisação
Qualidade Pragmática (KROGSTIE, 2013)	É a correspondência entre o meta-modelo e a interpretação do leitor e a aplicação dele. Refere-se à compreensão (como o meta-modelo foi entendido) e pode diferenciar entre qualidade pragmática social (pessoas) e qualidade pragmática técnica (ferramentas).	Compreensibilidade
Qualidade Organizacional (KROGSTIE, 2013)	Também chamado de qualidade deôntica, define se todas as declarações do meta-modelo contribuem para o cumprimento dos objetivos da modelagem, e se todos os objetivos de modelagem são abordados através do meta-modelo.	Flexibilidade, Integração e Implementabilidade

1.4.1.1 Diretrizes para Verificação de Qualidade de Meta-modelos

Para a avaliação das contribuições dessa pesquisa foi necessário estudar um conjunto de trabalhos que abordassem a verificação de critérios de qualidade em processos de meta-modelagem (KROGSTIE, 2012) e na avaliação de modelos conceituais (KROGSTIE, 2013; GARZA et al., 2016; KAHN; BATSON; SCHILLING, 2012; BATINI; SCANNAPIECO, 2006; MOODY; SHANKS, 2003). A maioria destes trabalhos é limitada. Muitas das abordagens estudadas apresentam simplesmente conjuntos de critérios de qualidade a serem avaliados mas não propõem as métricas e operacionalização necessárias para a avaliação. Operacionalização é a maneira de sair do conceito e se chegar ao indicador. Krogstie (KROGSTIE, 2012), por exemplo, não apresenta um conjunto de métricas nem definições para operacionalização de uma avaliação de meta-modelos. Para preencher esta lacuna, foi definido um conjunto de indicadores e operacionalização para o conjunto dos critérios de qualidade selecionados e consolidados a partir dos artigos científicos estudados (GARZA et al., 2016; KROGSTIE, 2013; KROGSTIE, 2012; KAHN; BATSON; SCHILLING, 2012; BATINI; SCANNAPIECO, 2006; MOODY; SHANKS, 2003). Isto originou a definição preliminar de um guia que apresenta, além dos indicadores e critérios, um conjunto de definições operacionais para verificação da qualidade de modelos. Nesse trabalho, este guia foi aplicado para verificação

da qualidade de meta-modelos no contexto de MDE.

O desenvolvimento dos critérios de qualidade estão detalhados na Tabela 2.

Para se calcular o valor dos indicadores de qualidade, foi escolhido o método estatístico de análise fatorial. A análise fatorial é uma técnica de redução de dados utilizada para reduzir uma grande quantidade de variáveis observadas a um número menor de fatores (KIM; MUELLER, 1978a; KIM; MUELLER, 1978b). Essa redução se baseia no padrão de correlação observado entre as variáveis, e é representada pelos fatores. Nesse trabalho, as variáveis observadas serão os critérios de qualidade para cada Indicador (fator). Para se chegar ao indicador de Qualidade Empírica, devem ser observados os critérios Compreensibilidade e Simplicidade, por exemplo. Assim, a análise fatorial pode ser utilizada para estimar um indicador que não é diretamente observável (qualidade, por exemplo), mas que representa a variação dos critérios observados. A execução desta análise estatística será demostrada no Capítulo 6.

Tabela 2 – Desenvolvimento e operacionalização dos critérios de qualidade. Adaptado de (GARZA et al., 2016; KROGSTIE, 2013; KAHN; BATSON; SCHILLING, 2012; BATINI; SCANNAPIECO, 2006; MOODY; SHANKS, 2003).

Critérios	**Definição Conceitual**	**Guia Operacional**	**Métricas Sugeridas**
Compreensibilidade (KROGSTIE, 2013; KAHN; BATSON; SCHILLING, 2012; MOODY; SHANKS, 2003)	Definido como a facilidade com a qual os elementos e estruturas do meta-modelo podem ser compreendidos.	Proponha um cenário/tarefa para observar a execução pelos usuários e avaliar os resultados das tarefas. Forneça uma escala para o usuário indicar o nível de compreensão. Selecione os problemas a serem resolvidos	M1 - Número de erros de interpretação de usuários (desenvolvedores e especialistas) M2 - Nível de classificação da compreensão pelos especialistas (rating). M3 - Nível de classificação da compreensão pelos desenvolvedores (rating).
Simplicidade (GARZA et al., 2016; KROGSTIE, 2013; KAHN; BATSON; SCHILLING, 2012; MOODY; SHANKS, 2003)	Define se o meta-modelo contém os elementos e relacionamentos mínimos possíveis ainda representando o domínio.	Meça quantitativamente os componentes/itens do meta-modelo. Estime qualitativamente (fácil, médio, difícil) a complexidade. Forneça uma escala para o usuário indicar o nível de complexidade.	M4 - Número de MetaClasses. M5 - Complexidade do Sistema (MetaClases+Relacionamentos). M6 - Complexidade Total (MetaClasses + Relacionamentos + Atributos). M7 - Nível de Complexidade.

Continua na próxima página

Tabela2 – Continuação da página anterior

Critérios	**Definição Conceitual**	**Definição Operacional**	**Métricas Sugeridas**
Corretude (KROGSTIE, 2013; KAHN; BATSON; SCHILLING, 2012; BATINI; SCANNAPIECO, 2006; MOODY; SHANKS, 2003)	Definido como a conformidade do meta-modelo ao formalismo representacional (regras e melhores práticas da técnica de modelagem). Se o meta-modelo é bem projetado.	Verifique se todas as expressões gramaticais usadas para criar o meta-modelo fazem parte da linguagem de modelagem (meta-meta-modelo).Geralmente, esse critério já é coberto quando são usadas ferramentas de modelagem (por exemplo, as ferramentas desenvolvidas com o Eclipse Modeling Framework).	M8 - Número de violações aos padrões da linguagem de modelagem.
Validade (definida em termos de integridade por (KROGSTIE, 2013))	Define se o meta-modelo apresenta de maneira correta (válida) todas as regras de negócios que se aplicam ao domínio.	Liste as necessidades (objetivos) do projeto e verifique quais associações no modelo correspondem às necessidades específicas do projeto.	M9 - Número de regras de negócio incorretas.
Integridade (GARZA et al., 2016; KROGSTIE, 2013; MOODY; SHANKS, 2003)	Define se as restrições e associações no meta-modelo correspondem às necessidades específicas do projeto. Entendida como a medida em que o modelo está em conformidade com as regras e processos de negócios para garantir a integridade dos dados do domínio e impõe relações significativas e restrições que mantêm a intenção do propósito original do domínio.	Liste as necessidades (objetivos) do projeto e verifique quais regras no modelo estão de acordo ou violam as necessidades específicas do domínio.	M10- Número de restrições adicionais especificadas no meta-modelo que não são requisitadas pelas regras de negócio. M11 - O meta-modelo provê acompanhamento de mudanças e informações de proveniência.
Completude (GARZA et al., 2016; KROGSTIE, 2013; KAHN; BATSON; SCHILLING, 2012; BATINI; SCANNAPIECO, 2006; MOODY; SHANKS, 2003)	Define se o meta-modelo contém todas as informações necessárias para suportar os requisitos do domínio e das funcionalidades do sistema.	Liste os objetos/elementos essenciais do domínio e verifique (de acordo com especialistas) se todos esses conceitos estão incluídos no meta-modelo.	M12 - Número de objetos ausentes. M13 - Número de objetos supérfluos.

Continua na próxima página

Tabela2 – Continuação da página anterior

Critérios	Definição Conceitual	Definição Operacional	Métricas Sugeridas
Integração (GARZA et al., 2016; KROGSTIE, 2013; KAHN; BATSON; SCHILLING, 2012; MOODY; SHANKS, 2003)	Define se o meta-modelo suporta terminologias controladas e a consistência do meta-modelo com necessidades corporativas ou sistemas existentes.	No caso dos domínios que contêm um vocabulário documentado, use-o como referência para verificar (M14) o uso desse vocabulário no meta-modelo. Identifique os objetos (elementos/conceitos) usados em outros sistemas que representam o domínio (necessidades corporativas) e verifique se o meta-modelo atende a esses elementos. Forneça uma escala para as partes interessadas (especialistas) indicarem o nível de integração com as necessidades corporativas.	M14 - Número de terminologias que correspondem ao vocabulário controlado. M15 - Número de conflitos com sistemas existentes. M16 - Número de itens duplicados em sistemas ou projetos existentes. M17 - Capacidade de integração aos sistemas ou necessidades corporativas (rating).
Reuso (KROGSTIE, 2013)	Define o nível de reuso do meta-modelo e esta relacionado a M16.	No caso de um número positivo para M16, verifique quais são reutilizados como parte do novo meta-modelo.	M18 - Número de itens existentes reusados.
Minimalização (BATINI; SCANNAPIECO, 2006)	Define se nenhum requisito é representado mais de uma vez.	Verificar os componentes/itens do meta-modelo para identificar redundância.	M19 - Número de MetaClasses redundantes. M20 - Número de Atributos redundantes.
Flexibilidade (GARZA et al., 2016; KROGSTIE, 2013; KAHN; BATSON; SCHILLING, 2012; MOODY; SHANKS, 2003)	Definido em termos de extensibilidade, adaptabilidade e escalabilidade. A facilidade em que o meta-modelo pode se adaptar às mudanças sem alterar a sua integridade.	Identifique possíveis mudanças no meta-modelo visando estendê-lo. Analise o impacto (custo e viabilidade) da mudança.	M21 - Número de itens do meta-modelo que estão sujeitos a alterações. M22 - Custo estimado da mudança (ver ajustes nas variáveis cobertas por M24, M25, M26).
Implementabilidade (GARZA et al., 2016; KROGSTIE, 2013; KAHN; BATSON; SCHILLING, 2012; MOODY; SHANKS, 2003)	Definido em termos de: (1) Aplicabilidade - diversidade de usos do meta-modelo. (2) Custo - a facilidade com que o meta-modelo pode ser implementado dentro dos recursos do projeto.	Meça quantitativamente o número de utilizações possíveis para aplicar o meta-modelo (usos acadêmicos ou industriais?, para criar bancos de dados ou desenvolvimento de software?). Estimar as principais variáveis que compõem o custo de desenvolvimento.	M23 - Número de usos pretendidos. M24 - Estimativa de tempo de programação. M25 - Tempo de treinamento. M26 - Custo de aquisição e/ou licença de hardware e software.

1.5 ORGANIZAÇÃO DA TESE

Este documento esta organizado como descrito a seguir:

- O Capítulo 2 - apresenta os conceitos relevantes relacionados ao contexto deste trabalho, e as tecnologias e ferramentas utilizadas para a construção da solução

proposta.

- O Capítulo 3 - apresenta o estado da arte e os trabalhos relacionados a esta pesquisa.
- O Capítulo 4 apresenta as principais contribuições definidas no escopo desse trabalho. Este apresenta as principais caraterísticas do domínio e o meta-modelo definido para representar dados biológicos moleculares. Apresenta também os algoritmos para identificação de esquemas e geração de bases de dados, e a arquitetura do domínio composta de todas as soluções propostas nesse trabalho.
- O Capítulo 5 - apresenta a implementação de referência do domínio para validação das soluções propostas nesse trabalho.
- O Capítulo 6 apresenta as análises empíricas realizadas para avaliação das soluções propostas nesse trabalho.
- O Capítulo 7 apresenta as considerações finais destacando o atendimento aos objetivos da pesquisa, as limitações e trabalhos futuros, e as publicações originadas desse trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

"If I have seen further it is by standing on the shoulders of Giants."

—Isaac Newton

Nesse capítulo serão apresentados os conceitos, teorias e abordagens relevantes para fundamentação do contexto ao qual este trabalho está inserido. Esta fundamentação teórica se faz necessária para a compreensão dos diversos fatores que podem influenciar o desenvolvimento do projeto. Estes conceitos foram classificados em cinco eixos principais: Engenharia Orientada a Modelos, que será apresentado na Seção 2.1; Dados Semi-estruturados, que será apresentado na Seção 2.2; Bases de Dados *Not Only SQL* (NoSQL), que será apresentado na Seção 2.3; Análise de Dados Biológicos Moleculares, que será apresentado na Seção 2.4; e Arquitetura Lambda, que será apresentado na Seção 2.5.

A Tabela 3 apresenta um guia metodológico resumindo as atividades executadas para identificação das evidências que serão discutidas nesse capítulo.

Etapa de Caracterização do Ambiente - Fundamentação teórica	
Objetivo	Compreender os fatores que podem influenciar o desenvolvimento do projeto
Atividades	Realizar um levantamento bibliográfico para fundamentar os principais conceitos e teorias aplicáveis a essa pesquisa.
Evidências	Fundamentação teórica; Categorização dos conceitos existentes; Descrição de diferentes visões, conceitos e abordagens.

Tabela 3 – Guia metodológico para a etapa de Caracterização do Ambiente (Fundamentação teórica)

2.1 ENGENHARIA ORIENTADA A MODELOS

Nas últimas décadas, foram propostas muitas técnicas e linguagens de modelagem para apoiar o projeto e o desenvolvimento de sistemas de software complexos. Muitas dessas linguagens foram definidas no contexto de abordagens metodológicas - como metodologias/processos estruturados, orientados a objetos ou unificados, fundamentalmente com o objetivo de facilitar e compartilhar uma visão comum e coerente do sistema em estudo e, consequentemente, facilitar a comunicação entre as partes interessadas (*stakeholders*) (BOOCH, 1994; WEAVER, 1993; YOURDON, 1989; KROGSTIE, 2012). No entanto, durante a última década, surgiu uma nova tendência de abordagens considerando modelos não apenas como artefatos de documentação, mas como artefatos centrais no processo de engenharia de software.

Além dos benefícios referidos acima, modelos também permitem - através de técnicas complexas como meta-modelagem, transformação de modelos, geração de código ou interpretação de modelos - a criação ou a execução automática de sistemas de software baseados nesses modelos. Essas abordagens - como Arquitetura Orientada a Modelos (em inglês, *Model-Driven Architecture* - MDA) (SIEGEL, 2014; FRANKEL, 2002), Fábricas de Software (GREENFIELD; SHORT, 2003) ou recentemente Engenharia de Linguagem Específica de Domínio(em inglês, *Domain Specifit Language* - DSL) (VOELTER et al., 2013) - foram classificadas genericamente como Engenharia Orientada a Modelos (em inglês, *Model-riven Engineering - MDE*), mas também por nomes relacionados, como a Engenharia Baseada em Modelos (em inglês, *Model-Based Engineering* - MBE), Desenvolvimento Orientado por Modelo (em inglês, *Model-Driven Development* - MDD), Desenvolvimento de Software Orientado por Modelo (em inglês, *Model-Driven Software Development* - MDSD) (MELLOR; BALCER, 2002; ATKINSON; KüHNE, 2003; STAHL; VOELTER; CZARNECKI, 2006; SELIC, 2008), Desenvolvimento Específico de Domínio (KELLY; TOLVANEN, 2008), Modernização Orientada por Modelos[1], ou Teste Baseado em Modelo (em inglês, *Model-Based test* - MBT) (UTTING; LEGEARD, 2007).

Em geral, estes paradigmas existentes em MDE compartilham os mesmos princípios básicos (BRAMBILLA; CABOT; WIMMER, 2012): (1) modelos são usados para representar aspectos de um sistema de software em algum nível de abstração; (2) os modelos são exemplos(instâncias) de meta-modelos (ou estão em conformidade com eles); (3) as transformações do modelo favorecem a geração de códigos no processo de desenvolvimento de software; e (4) modelos podem ser expressos por meio de linguagens de modelagem. Além disso, há quatro cenários principais em que as técnicas de MDE podem ser aplicadas (BRAMBILLA; CABOT; WIMMER, 2012): (1) construção de novas aplicações de software, (2) modernização de software (por exemplo, reengenharia de software), (3) uso de modelos em tempo de execução para representar o contexto ou executar o sistema, e (4) integração de ferramentas de software. Nesse trabalho, desenvolvemos soluções baseadas em modelos para lidar com problemas relacionados aos dois primeiros cenários: engenharia reversa para identificação de esquemas de bases de dados e criação de banco de dados a partir da transformação dos modelos extraídos para códigos de esquemas.

Estas abordagens compartilham conceitos e termos comuns que precisam ser abstraídos, discutidos e entendidos (SILVA, 2015). Vamos apresentar as definições para alguns desses conceitos essenciais de MDE: sistema, modelo, meta-modelo e suas relações.

Sistema - no contexto de MDE, adotamos a definição de sistema como um conceito abstrato para designar uma aplicação de software, uma plataforma de software ou outro artefato de software. Adicionalmente um sistema pode ser composto de outros subsistemas e um sistema pode ter relações com outros sistemas (um sistema pode se comunicar

[1] http://adm.omg.org/

com outros) (SILVA, 2015).

Modelo - Um modelo é uma abstração de um sistema em estudo (também conhecido como "Universo do Discurso"ou apenas "sistema"), que pode já existir ou se destina a existir no futuro. *Definição de Modelo*: No contexto de MDE a definição de modelo se diferencia do conceito definido no contexto de banco de dados para modelos de dados (Modelo Relacional, por exemplo) que tem o propósito de definir as regras e restrições estruturais em uma base de dados. Na ausência de uma definição comum para "modelo", é relevante referir algumas de suas tentativas populares: (1) modelo é um conjunto de declarações sobre o sistema em estudo (SEIDEWITZ, 2003); (2) modelo é uma abstração de um sistema (real ou baseado em linguagem) permitindo que previsões ou inferências sejam feitas (KUHNE, 2006); (3) modelo é uma representação reduzida de algum sistema que destaca as propriedades de interesse de um determinado ponto de vista (SELIC, 2003); (4) o modelo é uma simplificação de um sistema construído com o objetivo pretendido, de modo que um modelo deve ser capaz de responder a perguntas no lugar do sistema original (BEZIVIN et al., 2001); e (5) Modelo é um sistema que ajuda a definir e dar respostas do sistema em estudo sem a necessidade de considerá-lo diretamente (SILVA, 2015).

Meta-modelo - Há uma variedade de definições para o meta-modelo, algumas delas não são claras ou muito fracas, como uma definição da *Object Management Group* (OMG) que simplesmente afirma que "um meta-modelo é um modelo de modelos" (SIEGEL, 2014). No entanto, alguns autores refletiram sobre esses conceitos e dão as seguintes definições: (1) um meta-modelo é um modelo que define o idioma para expressar um modelo (SIEGEL, 2014); (2) um meta-modelo é um modelo de linguagem de modelos (FAVRE; NGUYEN, 2005); (3) um meta-modelo é um modelo de especificação para o qual os sistemas em estudo que estão sendo especificados são modelos em uma determinada linguagem de modelagem (SEIDEWITZ, 2003). No entanto, para este trabalho vamos utilizar a definição de Silva que define “Meta-modelo como um modelo que define a estrutura de uma linguagem de modelagem” (SILVA, 2015).

Plataforma - consiste em um conjunto integrado de elementos computacionais que permitem o desenvolvimento e a execução de uma classe de produtos de software (SILVA, 2015; VOELTER et al., 2013; GREENFIELD; SHORT, 2003; BRAMBILLA; CABOT; WIMMER, 2012).

2.1.1 Linguagens de Modelagem Específicas de Domínio

Modelos (ou seja, uma instância de um meta-modelo) são criados por meio de linguagens textuais ou visuais. Quando soluções de MDE são desenvolvidas, essas linguagens são necessárias para expressar os modelos a partir dos quais os artefatos de destino são

gerados. Essas linguagens são chamados de Linguagens de Modelagem Específicas de Domínio (*Domain Specific Modeling Languages* - DS(M)L), pois são projetados para resolver problemas em um domínio específico. No entanto, às vezes, as linguagens de modelagem podem ser aplicadas em qualquer domínio (Linguagens de Modelagem de Propósitos Gerais) (BRAMBILLA; CABOT; WIMMER, 2012). Algumas ferramentas estão disponíveis para definir DSMLs baseadas em meta-modelos, isto é, linguagens que permitem a criação de modelos cuja estrutura é determinada por um meta-modelo. Essas ferramentas facilitam a definição de uma notação (sintaxe concreta) para o meta-modelo (sintaxe abstrata), que pode ser textual ou gráfica (ou uma combinação de ambas). Moody (MOODY, 2010) definiu um conjunto de princípios para criação de notações visuais. Estes princípios serão discutidos na Seção 2.1.1.1.

Além da sintaxe abstrata e da sintaxe concreta, uma DSML tem a semântica como um terceiro elemento. A semântica define o comportamento da DSML. Esta semântica é tipicamente fornecida pela construção de um tradutor (ou seja, um compilador) para outra linguagem que já tenha uma semântica bem definida (por exemplo, uma linguagem de programação) ou, ainda, um interpretador (KLEPPE; WARMER; BAST, 2003).

O Meta-modelo e a notação são usados para gerar um editor de modelo (ou seja, uma ferramenta que cria a instância do meta-modelo a partir da notação visual ou textual). As ferramentas mais usadas para linguagens de domínio textuais são *Xtext*[2] e *Meta Programming System* (MPS)[3], e para linguagens gráficas *Sirius*[4] e *Metaedit*[5] . No caso de linguagens textuais, as ferramentas fornecem uma notação do tipo *Backus-Naur Form* (BNF) para expressar a gramática do idioma. Para linguagens gráficas, as ferramentas fornecem um editor gráfico para definir os símbolos dos elementos do meta-modelo e a paleta de ferramentas do editor gerado.

As linguagens específicas de domínio têm sido usadas desde os primeiros anos da programação, no entanto, a MDE aumentou o interesse nelas. Como mencionado anteriormente, a maioria das soluções de MDE envolve a definição de uma ou mais linguagens para que os usuários criem e/ou visualizem os modelos necessários. Uma estratégia possível para visualizar os modelos é definir uma notação usando uma ferramenta de definição de DSML.

Ainda, uma solução MDE pode envolver uma cadeia de transformações de modelos para gerar os artefatos de software desejados a partir dos modelos de origem. Três tipos de transformações de modelo são comumente usados: modelo para modelo (M2M), modelo para texto (M2T) e texto para modelo (T2M). A complexidade das transformações depende principalmente do nível de abstração dos meta-modelos (aos quais os modelos estão em conformidade).

[2] https://eclipse.org/Xtext/documentation/
[3] https://www.jetbrains.com/mps/
[4] https://eclipse.org/sirius/
[5] http://www.metacase.com/mep/

Sintaxe Visual (Concreta)	=	Símbolos Gráficos (Vocabulário Visual)	+	Regras de Composição (Gramática Visual)

2.1.1.1 Fundamentos para Definição de Notações Visuais

Uma boa notação deve ter eficácia cognitiva, que define a precisão, a facilidade e a velocidade com que a mente humana pode processar uma representação (MOODY, 2010). Segundo Moody (MOODY, 2010), a anatomia de uma boa notação visual consiste em:

A fim de fornecer uma base científica para a avaliação e concepção de notações visuais, algumas teorias básicas são necessárias, como a *Teoria Descritiva*, definindo que somente ao entender como e por que as notações visuais se comunicam, pode-se melhorar sua capacidade de comunicação. A descrição fornece a base para a prescrição. A *Teoria Prescritiva*, aponta que a definição de princípios explícitos transforma o design de notação visual de um processo inconsciente em um processo autoconsciente. A *Teoria Prescritiva* é a base para a construção da sintaxe concreta.

Os princípios propostos por Moody (MOODY, 2010) para construção de notações visuais efetivas sob a Teoria Prescritiva são: Claridade Semiótica, Discriminação Perceptiva, Transparência Semântica, Gerenciamento da Complexidade, Integração Cognitiva, Codificação Dupla, Economia Gráfica e Ajuste Cognitivo. A seguir serão apresentados detalhes sobre cada um destes princípios:

Claridade Semiótica - O princípio da claridade semiótica define que deve haver uma correspondência de 1:1 (um para um) entre os construtores semânticos (meta-modelo) e os símbolos gráficos da linguagem. As línguas naturais não são sistemas notacionais, pois contêm sinônimos e homônimos, mas muitas linguagens artificiais são (por exemplo, notação musical, notação matemática). Os requisitos de um sistema notacional restringem as expressões permitidas em uma linguagem para maximizar a precisão, a expressividade e a parcimônia, que são metas desejáveis de *design* para notações de engenharia de software. Quando não houver uma correspondência um-para-um entre construtores e símbolos, uma ou mais das seguintes anomalias podem ocorrer (MOODY, 2010):

- A *redundância de símbolo* ocorre quando vários símbolos gráficos podem ser usados para representar o mesmo construtor semântico.
- A *sobrecarga de símbolo* ocorre quando dois construtores diferentes podem ser representadas pelo mesmo símbolo gráfico.
- O *excesso de símbolo* ocorre quando os símbolos gráficos não correspondem a qualquer construtor semântico.
- O *déficit de símbolo* ocorre quando existem construtores semânticos que não são representadas por nenhum símbolo gráfico.

Discriminação Perceptiva - O princípio da discriminação perceptiva define a facilidade e a precisão em que os diferentes símbolos gráficos podem ser claramente diferenciados uns dos outros. Isso se relaciona com a primeira fase do processamento de informação visual humana: discriminação perceptiva. A discriminação precisa entre símbolos é um pré-requisito para interpretação precisa de diagramas. Esta discriminação é definida em termos de (MOODY, 2010):

- *Distância visual* entre os símbolos. Isso é medido pelo número de variáveis visuais, nas quais elas diferem, e o tamanho dessas diferenças (medido pelo número de etapas perceptíveis). Cada variável visual tem um número infinito de variações físicas, mas apenas um número finito de etapas perceptíveis (valores que são confiavelmente discrimináveis pela mente humana). Em geral, quanto maior a distância visual entre os símbolos, mais rápido e precisamente eles serão reconhecidos. Se as diferenças forem sutis demais, poderão ocorrer erros na interpretação. Muitas notações de engenharia de software têm uma discriminação muito fraca, já que consistem em formas e linhas de conexão que são muito semelhantes. Por exemplo, estudos experimentais mostram que entidades e relacionamentos são frequentemente confundidos em diagramas Entidade-Relacionamento (ER). A distância visual entre os símbolos é relativamente pequena, pois eles diferem em apenas uma variável visual (forma) e as formas pertencem à mesma família (quadriláteros).

- A *primazia da Forma*. De todas as variáveis visuais, a forma desempenha um papel especial na discriminação entre os símbolos, pois representa a base primária na qual se identificam objetos no mundo real. De fato, as teorias de reconhecimento de objetos diferem apenas na medida em que consideram que as representações de objetos se baseiam apenas na forma ou se outras características também estão envolvidas. Por este motivo, a forma deve ser usada como variável visual primária para distinguir entre diferentes construtores semânticos. Por exemplo, a discriminação dos diagramas ER poderia ser melhorada usando formas de famílias diferentes.

- *Codificação Redundante*. Redundância é uma técnica importante na teoria da comunicação para reduzir erros e neutralizar o ruído. A distância visual entre símbolos pode ser aumentada por codificação redundante: usando diversas variáveis visuais para distinguir entre elas. Como exemplo, a cor poderia ser usada para melhorar a discriminação entre entidades e relacionamentos em diagramas ER.

- *"Popout"perceptiva*. De acordo com a teoria de integração de características, elementos visuais com valores únicos para pelo menos uma variável visual podem ser detectados previamente e em paralelo através do campo visual. Esses elementos parecem “sair” de uma exibição sem esforço consciente. Por outro lado, elementos visuais que são diferenciados por combinações únicas de valores (conjunções) requerem pesquisa em série, que é muito mais lenta e propensa a erros. A clara implicação

disso para o *design* de notação visual é que cada símbolo gráfico deve ter um valor único em pelo menos uma variável visual.

- *Diferenciação textual.* Às vezes, as notações de engenharia de software baseiam-se em texto para diferenciação entre símbolos. Por exemplo, a UML usa frequentemente texto e características tipográficas (negrito, itálico, sublinhado) para distinguir entre os tipos de elementos e relacionamentos. Símbolos que diferem apenas nas características textuais são tecnicamente homógrafos, pois possuem distância visual zero.

Transparência Semântica - O princípio da transparência semântica define que devem ser utilizadas representações visuais em que sua aparência sugira o significado, ou seja, o significado de um símbolo pode ser inferido a partir de sua aparência. Embora a Discriminação Perceptiva simplesmente exija que os símbolos sejam diferentes uns dos outros, este princípio requer que eles fornecem pistas para o seu significado (forma implica conteúdo). O conceito de transparência semântica formaliza noções informais de "naturalidade" ou "intuitividade" que são frequentemente usadas quando se discutem notações visuais, pois podem ser avaliadas experimentalmente. Representações semanticamente transparentes reduzem a carga cognitiva porque elas têm mnemônicos embutidos: seu significado pode ser percebido diretamente ou facilmente aprendido (MOODY, 2010). Ícones são símbolos que se assemelham perceptivamente aos conceitos que representam, por exemplo.

Gerenciamento da Complexidade - O princípio da gerência de complexidade define que devem ser incluídos mecanismos explícitos para lidar com a complexidade. No contexto de notações visuais, "complexidade" refere-se à complexidade diagramática, que é medida pelo número de elementos (ocorrências de símbolos ou símbolos) em um diagrama. Embora isso seja aparentemente um problema de nível de diagrama, ele requer recursos de nível de notação para resolvê-lo. A complexidade tem um efeito importante sobre a eficácia cognitiva, pois a quantidade de informação que pode ser transmitida de forma eficaz por um único diagrama é limitada pelas capacidades perceptivas e cognitivas humanas (MOODY, 2010):

- Limites perceptivos: A capacidade de diferenciação entre elementos de um diagrama aumenta com o tamanho do diagrama.
- Limites cognitivos: O número de elementos de um diagrama que pode ser compreendido por vez é limitado pela capacidade de memória de trabalho. Quando isso é excedido, um estado de sobrecarga cognitiva se instala e a compreensão se degrada rapidamente.

Para representar efetivamente situações complexas, as notações visuais devem fornecer mecanismos de modularização e estruturação hierárquica. A *modularização* reduz a complexidade de grandes sistemas dividindo-os em partes ou subsistemas menores. A *hierar-*

quia permite que os sistemas sejam representados em diferentes níveis de detalhe, com complexidade gerenciável em cada nível (MOODY, 2010).

Integração Cognitiva - O princípio da integração cognitiva define que devem ser incluídos mecanismos explícitos para suportar a integração de informação entre os diferentes diagramas. Esta é uma questão crítica na egenharia de software, onde os problemas são tipicamente representados por sistemas de diagramas, em vez de diagramas simples. Aplica-se igualmente a diagramas do mesmo tipo (integração homogênea) ou diagramas de tipos diferentes (integração heterogênea), por exemplo, um conjunto de diagramas *Unified Modeling Language* (UML). Esse princípio está intimamente relacionado ao Gerenciamento de Complexidade, que leva a vários diagramas como resultado da modularização, mas se aplica mesmo quando a modularização não é usada (devido à integração heterogênea). Para que as representações de vários diagramas sejam cognitivamente eficazes, elas devem incluir mecanismos explícitos para apoiar (MOODY, 2010):

- Integração conceitual: mecanismos para ajudar o leitor a reunir informações de diagramas separados em uma representação mental coerente do sistema.
- Integração perceptiva: pistas perceptivas para simplificar a navegação e transições entre diagramas.

Expressividade Visual - O princípio da expressividade visual define que uma gama completa de variáveis e capacidades visuais devem ser utilizadas. Esta mede a utilização do espaço de design gráfico. Enquanto a distância visual (Discriminação Perceptual) mede a variação visual entre símbolos, a expressividade visual mede a variação visual em todo o vocabulário visual. O uso de uma variedade de variáveis visuais (por exemplo, forma, tamanho, cor, brilho, orientação e textura) resulta em uma representação perceptivelmente enriquecida que explora vários canais de comunicação visual e maximiza o descarregamento computacional. Expressividade visual particiona o conjunto de variáveis visuais em dois subconjuntos (MOODY, 2010):

- Variáveis transportadoras de informação: variáveis usadas para codificar informações em uma notação.
- Variáveis livres: variáveis não usadas (formalmente).

A maioria das notações em Engenharia de Software usa apenas uma única variável para codificar informação: a forma.

Codificação Dupla - O princípio da codificação dupla define que elementos de texto podem ser utilizados para complementar os elementos gráficos. A Discriminabilidade Perceptual e a Expressividade Visual aconselham contra o uso de texto para codificar informações em notações visuais. No entanto, isso não significa que o texto não possa ser

usado *design* de notação visual. Imagens e palavras não devem ser mutuamente exclusivas. De acordo com a teoria de codificação dupla, usar texto e gráficos juntos para transmitir informações é mais eficaz do que usar apenas um destes. Quando a informação é apresentada verbalmente e visualmente, as representações dessa informação são codificadas em sistemas separados na memória de trabalho e as conexões referenciais entre as duas são reforçadas. Isso sugere que a codificação textual é mais eficaz quando usada em uma função de suporte: complementar em vez de substituir os gráficos (MOODY, 2010).

Economia Gráfica - O princípio da economia gráfica define que o número de diferentes símbolos gráficos devem ser cognitivamente gerenciáveis. A complexidade gráfica é definida pelo número de símbolos gráficos em uma notação: o tamanho de seu vocabulário visual. Isso difere da complexidade diagramática (Gerenciamento de Complexidade), pois está relacionada à complexidade no nível de tipo (linguagem) e não ao nível de sentença. As notações em engenharia de software tendem a aumentar a complexidade gráfica ao longo do tempo, principalmente devido aos esforços para aumentar sua expressividade semântica. Cada novo construtor normalmente requer um novo símbolo e, embora novos construtores sejam frequentemente adicionadas, os antigos raramente são removidos. Por exemplo, a primeira notação de fluxogramas de programas foi iniciada com 5 símbolos, mas expandida para 8 no momento em que o padrão ISO final foi publicado. Existem três estratégias principais para lidar com a complexidade gráfica excessiva (MOODY, 2010):

- Reduzir a complexidade semântica;
- Introduzir déficit de símbolo; e
- Aumentar a expressividade visual.

Ajuste Cognitivo - O princípio do ajuste cognitivo define que devem ser usados diferentes dialetos visuais para diferentes tarefas e audiências. A teoria do ajuste cognitivo é uma teoria amplamente aceita no campo dos sistemas de informação que foi validada em uma ampla gama de domínios, desde a tomada de decisões gerenciais até a manutenção de programas. A teoria afirma que diferentes representações de informação são adequadas para diferentes tarefas e diferentes públicos. O desempenho da resolução de problemas (que corresponde aproximadamente à eficácia cognitiva) é determinado por um ajuste de três vias entre: a representação do problema, as características da tarefa e as habilidades do solucionador de problemas. A maioria das notações de engenharia de software exibe um monolinguismo visual: elas usam uma única representação visual para todos os propósitos. No entanto, a teoria do ajuste cognitivo sugere que essa suposição de "tamanho único" pode ser inadequada e que diferentes dialetos visuais podem ser necessários. Estes representam dialetos visuais complementares em vez de concorrentes, conforme discutido anteriormente. Existem pelo menos duas razões para criar múltiplos dialetos visuais em um contexto engenharia de software:

- Diferenças entre especialista e novatos (habilidades de solucionador problemas) e
- Meio representacional (características da tarefa).

2.2 DADOS SEMI-ESTRUTURADOS

Em meados da década de 1990, surgiu a noção de dados semi-estruturados para definir as propriedades dos dados envolvidos na Web e as novas aplicações que surgiram em torno dela (por exemplo, bancos de dados de genoma ou bancos de dados geográficos), bem como lidar com a integração de fontes de dados independentes e navegação de dados (ou seja, para escrever consultas de dados sem conhecimento do esquema) (ABITEBOUL, 1996; ABITEBOUL; BUNEMAN; SUCIU, 2000; BUNEMAN, 1997).

Os dados semi-estruturados são caracterizados principalmente pelo fato de sua estrutura não estar definida em um esquema separado, mas está implícita nos próprios dados. Geralmente, os dados semi-estruturados são descritos como uma tupla de pares $Chave - Valor$. Chaves (campos e *tags*) denotam propriedades ou atributos dos dados, e os valores podem ser primitivos (isto é, valores atômicos de tipos como números, cadeias de caracteres e booleanos) ou complexos (por exemplo, *arrays*).

Dados semi-estruturados também possuem uma estrutura hierárquica (ou seja, estrutura aninhada) com um atributo raiz (ou pai) que pode incluir outros atributos aninhados.Vale a pena notar que os dados semi-estruturados representam um formato para expressar objetos incorporados ou agregados, mas sem a necessidade de definição prévia de um esquema.

Como indicado em (BUNEMAN, 1997), uma parte dos dados semi-estruturados pode ser formalizada com estruturas semelhantes a um grafo ou uma árvore. Se referências entre partes de dados são permitidas, a estrutura seria um grafo. Uma representação em grafo pode incluir nós de folha rotulados com dados significativos (valores atômicos) e arestas intermediárias rotuladas com símbolos que denotem os nomes dos atributos. Um nó raiz ou intermediário tem um nó filho para cada campo de objeto associado (aninhado).

A criação de dados em tempo de execução e persistentes geralmente requer a existência de uma especificação de sua estrutura. Por exemplo, objetos são instanciados de classes (ou seja, tipos) durante a execução de programas orientados a objeto, e um esquema deve ser definido antes de armazenar dados em um banco de dados relacional. No entanto, os dados semi-estruturados podem ser criados diretamente, porque incluem informações que descrevem sua estrutura. Por este motivo, estes tipos de dados são comumente caracterizados como "sem esquema fixo/rígido"e "auto-descritivo", pois as informações sobre a estrutura estão dentro dos dados mas nenhum esquema ou tipos foram definidos previamente em uma especificação separada dos dados.

A falta de esquemas explicitamente definidos (sem esquema) oferece alguns benefícios significativos aos desenvolvedores de aplicativos de banco de dados, como observado em

(ABITEBOUL, 1996; SADALAGE; FOWLER, 2012). Em particular, essa característica facilita a evolução dos dados e o uso de dados não uniformizados.

Evolução dos dados - A estrutura dos dados pode evoluir com frequência, pois alterações de esquema não são necessárias. Dados com a nova estrutura podem ser armazenados sem qualquer tipo de restrição. Em sistemas relacionais, a evolução dos dados requer a mudança do esquema e algum tipo de migração de dados.

Estrutura variada - Um dos principais pontos fortes dos dados semi-estruturados é permitir a variação da estrutura em dados do mesmo tipo. Por exemplo, uma propriedade pode ter valores de tipos diferentes ou pode ser opcional. Essas variações geralmente impactam em pequenas alterações na representação.

Embora a ausência de esquema forneça uma maior flexibilidade no gerenciamento de dados, ela pode apresentar algumas desvantagens para desenvolvedores. Quando um esquema é formalmente definido (por exemplo, um esquema relacional), é assegurado que somente os dados que se ajustam ao esquema podem ser manipulados no código do aplicativo, e os erros cometidos pelos desenvolvedores ao escrever o código são previamente identificados. Esta é a mesma analogia das linguagens estáticas e dinâmicas, que é comumente usada para observar a diferença entre dados semi-estruturados e dados que estão de acordo com um esquema rígido pré-definido (BUNEMAN, 1997). Quando um esquema não é pré-definido, erros como propriedades duplicadas ou ausentes não são capturados no momento em que os dados são criados.

A maioria dos bancos de dados NoSQL não tem esquema, por motivos como: (1) armazenam dados cuja estrutura está mudando rapidamente ou é tomada como desconhecida, e (2) o suporte ao desenvolvimento ágil é um dos principais requisitos que devem satisfazer. Novos domínios de aplicações nas quais é esperado que as estruturas de dados passem por mudanças rápidas, já foram identificados há vinte anos nos primeiros trabalhos sobre dados semi-estruturados (ABITEBOUL, 1996; BUNEMAN, 1997). O número dessas aplicações cresceu consideravelmente desde então.

XML tem sido o formato comumente usado para armazenar dados semi-estruturados a partir do advento da *Web* e sua adoção pelo *World Wide Web Consortium* (W3C) como um padrão para intercâmbio de dados na *Web* [6]. No entanto, desde a aparição do *JavaScript Object Notation* (JSON) no final da última década, o XML está perdendo predominância como formato de intercâmbio de dados em favor do JSON (SEVILLA; FELICIANO; GARCíA-MOLINA, 2017). JSON é o formato usado para armazenar dados em alguns dos bancos de dados NoSQL (MongoDB, por exemplo). Nestes, os dados são internamente codificados em algum formato de serialização binária, como o BSON [7]. Mais detalhes sobre bases de dados NoSQL e o formato JSON serão apresentados a seguir.

[6] https://www.w3.org/standards/xml/

[7] https://www.mongodb.com/json-and-bson

2.2.1 O Formato JSON

JSON[8] é um formato de texto amplamente utilizado para representar dados semi-estruturados. Essa notação está tomando o lugar do XML como formato de intercâmbio de dados devido a alguns benefícios como sua performance, simplicidade e facilidade de leitura (GOYAL; SINGH; RAMKUMAR, 2017). O JSON é um subconjunto da linguagem *JavaScript*, mais especificamente os dados JSON são literais *JavaScript*. Isso contribuiu significativamente para o sucesso do JSON. Existem dois padrões de JSON: ECMA e RFC7159. A principal diferença é que o ECMA considera qualquer valor JSON como um texto JSON válido e o RFC7159 estabelece que um texto JSON válido deve ter um objeto ou *array* como *root*. O JSON é altamente interoperável porque os dados trocados são simplesmente texto Unicode. A gramática do JSON especifica que um texto (também conhecido como documento) pode ser (1) um objeto formado por um conjunto de pares de chave-valor ou (2) um *array* (ou seja, uma lista ordenada) de valores. O tipo de um valor JSON pode ser um tipo primitivo (*Number, String ou Boolean*), um objeto ou um *array* de valores. *null* é usado para indicar que uma chave não tem valor. Esta gramática permite representar dados na forma de estruturas aninhadas usando objetos e arrays como construções de composição. É importante observar que JSON é uma notação destinada a expressar dados semi-estruturados em forma de árvore e sem esquema.

Esquemas JSON - A iniciativa *JSON Schema*[9] surgiu recentemente fornecendo especificações para descrição de esquemas JSON. Embora sua adoção ainda seja muito limitada, algumas ferramentas (por exemplo, validadores, geradores de esquemas, geradores de documentação) evidenciaram a utilidade de se ter esquemas JSON. Como exemplo de especificações podemos citar a indicação do tipo JSON de cada campo. Para um campo que hospeda um objeto, os campos ("propriedades do objeto") são especificados. Para um campo Array, o tipo dos itens é especificado e as propriedades necessárias também são especificadas.

2.3 BANCOS DE DADOS NoSQL

O termo NoSQL é usado para se referir a diferentes paradigmas de novos bancos de dados que são uma alternativa aos SGBDs relacionais. Aplicativos da Web, como redes sociais (por exemplo, *Facebook*), pesquisa de texto (por exemplo, *Google*) e comércio eletrônico (por exemplo, *Amazon*), que gerenciam dados muito volumosos e complexos, são alguns exemplos de cenários em que diferentes sistemas NoSQL foram usados com sucesso. A principal diferença entre Bancos de dados NoSQL e bancos de dados relacionais é o conjunto de propriedades que eles fornecem; enquanto bancos de dados relacionais fornecem

[8] http://www.w3schools.com/js/js$_j$*son*
[9] http://json-schema.org/

todas as propriedades do ACID (Atomicidade, Consistência, Isolamento e Durabilidade), bancos de dados NoSQL fornecem um subconjunto das propriedades do teorema CAP (*Consistency, Availability, Partition Tolerance*): Consistência (sempre que um nó de escrita é atualizado, todos os nós de leitura têm seus valores atualizados), Disponibilidade (o sistema opera continuamente mesmo quando partes dele são interrompidas) e Tolerância à Partição (o sistema lida com adição e remoção dinâmica de nós) e as propriedades BASE(*Basicaly Available, Soft-State, Eventualy Consistent*): basicamente disponível, funciona em todo o tempo; estado leve, não tem que ser consistente todo o tempo; e eventualmente consistente, o sistema torna-se consistente no momento devido (SADALAGE; FOWLER, 2012).

Os bancos NoSQL fornecem um conjunto variado de paradigmas de modelagem de dados destinados a gerenciar dados semi-estruturados e não estruturados. A maioria deles tem algumas propriedades comuns, por exemplo: não usam a linguagem SQL como padrão, os esquemas não precisam ser previamente definidos para especificar a estrutura de dados, a execução em *clusters* é o principal fator que determina seu design e eles são desenvolvidos como iniciativas de código aberto. As principais categorias de bancos de dados NoSQL são: orientados a documento, família de colunas (também conhecido como, orientado a colunas), chave-valor e bancos de dados baseados em grafos (SADALAGE; FOWLER, 2012).

Bancos de dados Chave-Valor - Fornecem a maneira mais simples de armazenar dados: um par chave-valor. O banco de dados é, portanto, um conjunto de pares chave-valor. A maioria dos sistemas chave-valor não assume nenhuma estrutura nos dados (por exemplo, *Riak*[10]), mas alguns deles permitem atribuir um tipo de dado a um valor (por exemplo, *Redis*[11]).

Bancos de dados Orientados a Documentos - Os dados também são armazenados como um conjunto de pares de chave-valor, mas o valor toma a forma de um documento estruturado (normalmente um documento semelhante a JSON) que pode ser navegado para obter dados específicos ou para formar consultas. Além disso, o banco de dados geralmente é organizado em um conjunto de coleções e cada coleção contém os documentos armazenados para um tipo de entidade. Consultas podem ser emitidas para coleções. *MongoDB*[12] e *CouchDB*[13] são os bancos de documentos mais utilizados.

Bancos de dados de Famílias de Colunas - Estes são organizados como uma coleção de linhas, cada uma delas consistindo de uma linha chave e um conjunto de famílias de

[10] http://basho.com/products/riak-kv/
[11] https://redis.io/
[12] https://www.mongodb.com/
[13] http://couchdb.apache.org/

colunas, cada uma delas, por sua vez, consistindo de um conjunto de pares chave-valor. *Cassandra* [14] e *Hbase*[15] são os sistemas de família de colunas mais utilizados.

Bancos de Dados Baseados em Grafos - Nestes, um banco de dados é armazenado como um grafo onde os nós contêm as propriedades da entidade e as arestas representam os relacionamentos entre as entidades. As arestas também podem ser rotuladas com propriedades. Ao contrário de outros sistemas NoSQL, os relacionamentos são a informação mais importante em bancos de dados de grafo. *Neo4J*[16] e *OrientDB*[17] são exemplos de bancos de dados de grafo.

Enquanto os bancos de dados de grafo são baseados em um modelo de dados que enfatiza os relacionamentos entre os dados de entidades, as outras três categorias são baseadas em um modelo de dados orientado a agregados. O modelo de dados orientado a agregado é um tipo de modelo que enfatiza a estrutura aninhada de uma entidade. No modelo orientado a agregados, as entidades têm propriedades cujo valor também podem ser conjuntos de propriedades e até entidades completas (uma estrutura semelhante a árvore de dados semi-estruturados) (SADALAGE; FOWLER, 2012). Objetos agregados (isto é, entidades aninhadas) geralmente são preferidos em substituição a referências entre objetos. Isto se dá pelo fato de os dados serem distribuídos por meio de *clusters* para obter escalabilidade, e referências a objetos poderiam envolver o contato com nós remotos.

Um esquema de um modelo de dados orientado a agregados para bancos de dados NoSQL também pode ser representado como uma árvore. Este seria basicamente formado por um conjunto de entidades conectadas por meio de dois tipos de relacionamentos: agregação e referência. Cada entidade pode ter uma ou várias propriedades especificadas por seu nome e seu tipo de dados. As entidades podem ser dos tipos raiz ou aninhadas. Uma entidade raiz não está aninhada em nenhuma outra entidade e uma entidade aninhada é aquela incorporada (ou embutida) em uma entidade raiz. A existência de versões de entidades significa que cada entidade possui tantos esquemas quanto versões e diferentes tipos de esquemas podem ser definidos para modelos de dados orientados a agregados. Nesse trabalho vamos apresentar os diferentes tipos de esquemas para bases de dados de variações de genoma. Esses dados são em sua maioria disponibilizadas como arquivos XML e também podem ser representadas como modelos de dados orientados a agregados. Estes esquemas serão detalhados no Capítulo 4.

[14] http://cassandra.apache.org/
[15] https://hbase.apache.org/
[16] https://neo4j.com/
[17] http://orientdb.com/orientdb/

2.4 ANÁLISE DE DADOS BIOLÓGICOS MOLECULARES

A análise de dados biológicos moleculares é orquestrada por um conjunto pré-determinado de passos, comumente chamados de "*workflow*" ou "*pipeline*" que vai desde a captura dos dados genéticos de um indivíduo, passa por processos de transformação e mineração e finaliza com o cruzamento (também conhecido como "*matching*") das variantes encontradas no indivíduo com variantes clínicas armazenadas em mais de 10 bancos de dados, para finalmente ser filtrada, ordenada e organizada para ser entregue ao especialista (biólogo, médico geneticista ou biomédico) (PABINGER et al., 2014).

Em resumo, a análise dos dados biológicos moleculares (especificamente dados genéticos) pode ser decomposta em quatro etapas distintas: (1) avaliação de qualidade dos dados brutos, (2) alinhamento de leitura a um genoma de referência, (3) identificação da variante, (4) anotação das variantes e visualização de dados (OIKAWA et al., 2004).

Na primeira etapa, os dados de entrada são os fragmentos de DNA (*reads*) produzidos por uma plataforma de sequenciamento, por exemplo as máquinas *Next-generation sequencing* (NGS). Nesta etapa é realizada uma análise de qualidade (*Quality Control* - QC) dos dados, que pode ser considerado um primeiro filtro de análise antes de passar para as próximas etapas. Os resultados dessa etapa são comumente apresentados em planilhas digitais para avaliação do bioinformata que realiza filtragem de *reads*, *scores* de qualidade e outras métricas.

A segunda etapa é mapear os *reads* contra um genoma de referência. Esta etapa é conhecida como alinhamento e para esta são utilizados pacotes de softwares específicos como BWA (*Burrows-Wheeler Aligner*)[18] e Bowtie[19], por exemplo. A escolha envolve quesitos de performance/eficiência e acurácia, uma vez que um genoma humano completo quando sequenciado pode ter mais de 30GB em sequências de DNA.

A terceira etapa inclui um processamento para conversão dos dados em um formato binário (BAM). Este formato permite indexar e armazenar as sequências de forma otimizada para leitura por softwares de visualização de sequência de DNA. Nesta etapa também acontece a identificação de variantes, que é responsável por ler as sequências e cruzar com bases de mutações para localizar trocas de bases, inserções ou deleções apresentadas nas sequências. A saída desta etapa consiste em um arquivo *variant call format* (VCF) que contém as variantes encontradas.

A quarta etapa consiste em cruzar o arquivo de variantes com bases que possuem as principais referências científicas de medicina genômica. A principal tarefa desta etapa é a adição de vários metadados às variantes detectadas. Entre estes metadados estão: a posição cromossômica, o efeito funcional sobre a proteína e outros dados. São estas referências que indicam se as alterações (variantes) presentes são ou não causadoras de uma doença.

[18] http://bio-bwa.sourceforge.net
[19] http://bowtie-bio.sourceforge.net/index.shtml

Como exemplo das bases consultadas nesse processo pode-se citar *OMIM*, *PubMed*, *Clinvar*, 1000 *Genomes Project*, ESP 6500 *exomes* , ESP 5400 *exomes*. Estas são disponibilizadas em variados formatos mas em geral são publicadas em arquivos texto ou semi-estruturados (XML). Além destes diferentes formatos, estas bases também apresentam variações na estrutura dos dados (esquemas).

Com a necessidade de consulta a essas bases, essa é considerada a etapa mais laboriosa, e que exige mais tempo. Ainda mais pelo fato de que todos estes dados são comumente reunidos e formatados em uma planilha digital. Estas planilhas são disponibilizadas para análise pelos especialistas que irão montar filtros para as seleções e combinações de variantes necessárias para escrita das conclusões e confecção de laudos.

Diante deste cenário, essa pesquisa identificou a necessidade de definir uma representação formal que dê suporte a este processo melhorando a análise de dados biológicos moleculares, especificamente na etapa de anotação de variantes.

2.5 A ARQUITETURA LAMBDA

Lambda (MARZ, 2013) é uma arquitetura genérica de processamento de dados projetada para lidar com grandes volumes de dados tanto no método *batch* quanto no de *stream-processing*. A Arquitetura Lambda se propõe a resolver o problema de computar funções arbitrárias em dados arbitrários em tempo real. A sua estrutura é decomposta em três camadas: "*batch layer*", que é responsável por persistir os dados (em um banco de dados NoSQL ou em um sistema de arquivos distribuídos); "*serving layer*", é responsável por realizar análises ou *views* sobre os dados persistidos e disponibilizá-las através de visões distintas; e "*speed layer*", que cria análises em tempo real. Todas essas camadas podem receber consultas da aplicação final e os dados presentes nelas podem ainda ser computados, cruzados ou agregados. A arquitetura Lambda foi desenvolvida para lidar com dados no contexto de BigData. Para um melhor entendimento, este conceito e suas principais características serão apresentadas na Seção 2.5.1. Em seguida serão discutidas as camadas da arquiteura Lambda.

2.5.1 BigData

Big Data (SAGIROGLU; SINANC, 2013) é um termo para crescentes conjuntos de dados massivos, variados e de estrutura complexa com dificuldades de armazenamento, análises e visualização de novos processos ou resultados.

De fato, o sequenciamento do genoma é uma aplicação pioneira do BigData. Um único genoma humano consiste em cerca de 3 bilhões de pares de bases de DNA. As bases de dados de sequências de genoma crescem constantemente. Um exemplo destas

bases é o *GenBank*[20], uma das bases mantidas pelo NIH (EUA), que consiste em uma coleção anotada de todas as seqüências de DNA publicamente disponíveis. Atualmente, o *Genbank* dobra de tamanho a cada 18 meses. Esta tendência foi confirmada em anos anteriores e deve ser mantida nos próximos anos. A caracterização do genoma humano em larga escala envolve a geração e interpretação de um enorme volume de dados em uma escala sem precedentes, e um dos benefícios potenciais é a medicina personalizada para apoiar as decisões clínicas. Isto pode impactar diretamente em diagnósticos de pacientes com câncer, por exemplo (CHIN; ANDERSEN; FUTREAL, 2011).

Para atingir os eixos principais (volume, velocidade, variedade e veracidade) envolvidos em Big Data é preciso ser capaz de lidar não só com um grande volume de dados oriundos de diversas fontes, mas também ser capaz de realizar operações em uma velocidade que seja próxima a tempo real. Calcular funções arbitrárias em um conjunto de dados arbitrário em tempo real é um problema difícil e não há ferramenta única que ofereça uma solução completa. Em vez disso, é necessário utilizar uma variedade de ferramentas e técnicas para construir um sistema completo (MARZ, 2013). É importante mencionar que esse trabalho não tem o objetivo de desenvolver uma solução para processamento de dados em tempo real, mesmo os dados e aplicações tratados no contexto dessa pesquisa serem pioneiros em Big Data. Para atender a este requisito, será feiro o uso da Arquitetura Lambda. A Arquitetura Lambda é uma abordagem genérica de processamento de dados projetada para lidar com grandes volumes de dados tanto no modo "*batch*"quanto no de "*stream-processing*".

2.5.2 Batch layer

A parte da Arquitetura Lambda que pré-computa as visões *batch* é chamada de "*batch layer*" (MARZ, 2013). A camada *batch* armazena a cópia mestre dos conjuntos de dados e pré-computa as visões *batch* nesse conjunto de dados mestre. O conjunto de dados mestre pode ser pensado como uma lista muito grande de registros. A camada *batch* precisa ser capaz de realizar duas operações: armazenar um imutável conjunto de dados mestre em constante crescimento, e calcular funções arbitrárias sobre esse conjunto de dados. Para pré-computar visões sobre um conjunto de dados, é preciso ser capaz de fazê-lo para qualquer visão e qualquer conjunto de dados. Há uma classe de sistemas chamados "sistemas de processamento em lote (*batch*)"que são construídos para realizar exatamente o que a camada de *batch* requer. Eles armazenam imutáveis conjuntos de dados em constante crescimento, e expõem primitivas computacionais para permitir calcular funções arbitrárias sobre esses conjuntos de dados. *Hadoop* (WHITE, 2010) é o exemplo clássico de um sistema de processamento em lote.

A forma mais simples da camada *Batch* pode ser representada em um pseudo-código como segue:

[20] http://www.ncbi.nlm.nih.gov/genbank/statisticshttp://www.ncbi.nlm.nih.gov/genbank/statistics

```
function runBatchLayer():
    while(true):
        recomputeBatchViews()
```

A camada *batch* é executada em laço *while* (*true*) e continuamente re-computa as visões de *batch*. Cálculos em lote (*batch*) são escritos como programas *single-threaded*, e ainda paralelizam em um *cluster* de máquinas. Este paralelismo implícito faz os cálculos da camada *batch* escalarem para conjuntos de dados de qualquer tamanho. Abaixo segue um exemplo de um cálculo na camada *batch* para demostrar um programa inerentemente paralelo.

```
Pipe pipe = new Pipe("counter");
pipe = new GroupBy(pipe, new Fields("url"));
pipe = new Every(
    pipe,
    new Count(new Fields("count")),
    new Fields("url", "count"));
Flow flow = new FlowConnector().connect(
        new Hfs(new TextLine(new Fields("url")), srcDir),
        new StdoutTap(),
        pipe);
flow.complete();
```

Esse código computa o número de exibição de páginas (*pageviews*) para cada *Uniform Resource Locator* (URL) dado um conjunto de dados de entrada de *pageviews* brutas. Nesse código todos os desafios de concorrência de agendamento, *merge* de resultados, e lidar com falhas de tempo de execução (como máquinas caindo) é feito para o usuário. Com o algoritmo escrito desta forma, ele pode ser distribuído em um *cluster MapReduce*, escalando para muitos nós que o usuário tenha disponível. Logo, se o usuário possuir 10 nós em seu *cluster MapReduce*, o cálculo terminará acima de 10 vezes mais rápido do que se ele tiver apenas um nó. No final da computação, o diretório de saída irá conter um certo número de arquivos com o resultado (MARZ, 2013).

2.5.3 Serving layer

A camada *batch* emite visões como o resultado de suas funções. Esta camada carrega as visões em um lugar em que elas possam ser consultadas. Esta tarefa é realizada pela camada de serviço ("*serving Layer*") (MARZ, 2013). Por exemplo, a camada *batch* pode pré computar uma visão contendo a contagem de exibição de páginas (*pageview*) para todo par $[url, hora]$. A visão *batch* é essencialmente apenas um conjunto de arquivos *flat* porém, não há nenhuma maneira de obter rapidamente o valor para uma determinada URL fora dessa saída. A camada de serviço indexa as visões da camada *batch* e as carrega para que possam ser eficientemente consultas para obter valores específicos. A camada de serviço é um banco de dados distribuído especializado que carrega uma visão *batch*,

a torna consultável, e continuamente troca em novas versões da visão *bacth* visto que elas são computadas pela camada *batch*. A camada *batch* geralmente gasta pelo menos algumas horas para fazer uma atualização, já a camada de serviço é atualizada em poucas horas (MARZ, 2013).

Um banco de dados da camada de serviço requer apenas atualizações em lote e leituras aleatórias (randômicas). Ele não precisa apoiar escritas aleatórias. Como escritas aleatórias causam a maior parte da complexidade em bancos de dados, ao não apoiar escritas aleatórias, os bancos de dados da camada de serviço podem ser mais simples. Esta simplicidade torna-os robustos, previsíveis, fáceis de configurar e fáceis de operar.

2.5.4 Speed layer

A camada de serviço atualiza sempre que a camada *batch* termina de pre-computar uma visão *batch*. Isto significa que os únicos dados não representados nas visões *batch* são os dados originados enquanto a pre-computação estava sendo executada. A tarefa restante para ter um sistema de dados totalmente em tempo real - ou seja, funções arbitrárias computadas em dados arbitrários em tempo real - é compensar essas últimas horas de dados. Este é o propósito da camada *speed* (*Speed layer*) (MARZ, 2013). Pode-se pensar a camada *speed* de maneira semelhante à camada *batch*, que produz visões com base nos dados que recebe, embora existam algumas diferenças fundamentais. Uma grande diferença é que para alcançar rápidas latências, a camada *speed* não considera todos os novos dados ao mesmo tempo. Ela atualiza a visão de tempo real à medida que recebe novos dados em vez de re-computar as visões como a camada de *batch*. Isso é chamado de "atualizações incrementais"em oposição a "atualizações re-computadas". Outra grande diferença é que a camada *speed* só produz visões em dados recentes, enquanto que a camada *batch* produz visões sobre todo o conjunto de dados. Considerando o exemplo de computar o número de *pageviews* para uma *url* ao longo de um intervalo de tempo, a camada *speed* precisa compensar *pageviews* que não foram incorporados nas visões *batch*. Isto será equivalente a algumas horas de *pageviews*. Como a camada *batch*, a camada *speed* mantém uma visão a partir de uma chave $[url, hora]$ para uma contagem de *pageviews*. Ao contrário da camada *batch*, que re-computa aquele mapeamento a cada tempo, a camada *speed* altera a sua visão à medida que recebe novos dados. Quando recebe uma nova *pageview*, ela incrementa a contagem para o correspondente $[url, hora]$ no banco de dados. A camada *speed* requer bancos de dados que suportam leituras e escritas aleatórias. Por suportar escritas aleatórias, esses bancos de dados são de ordens de magnitude mais complexas do que os bancos de dados usados na camada de serviço, tanto em termos de implementação quanto de operação.

A vantagem da arquitetura Lambda é que pode-se descartar partes que não são mais necessárias nas visões em tempo real. Este é um importante benefício, uma vez que a camada *speed* é muito mais complexa do que a camada *batch* e a camada de serviço. Esta

propriedade da Arquitetura Lambda é chamada de "isolamento de complexidade", o que significa que a complexidade é empurrada para uma camada cujo resultados são apenas temporários. Se algum problema ocorrer, pode-se descartar o estado de toda a camada *speed* e tudo voltará ao estado normal dentro de algumas horas. Esta propriedade limita consideravelmente o potencial impacto negativo da complexidade da camada *speed*.

A última parte da arquitetura Lambda reuni os resultados das visões *batch* e em tempo real para executar as funções de consulta. Para o exemplo de exibição de página, obtém-se os valores de contagem para o maior número de horas possível no intervalo das visões *batch*. Então, se consulta a exibição em tempo real para obter os valores de contagem para as horas restantes. Soma-se, então, todas as contagens individuais para obter o número total de páginas vistas durante esse intervalo. Um esforço precisa ser realizado para obter a sincronização certa entre as visões *batch* e as em tempo real.

Nas Seções anteriores foram cobertos vários conceitos da arquitetura lambda. Na Seção seguinte consta um resumo da arquitetura lambda para demostrar como esta funciona.

2.5.5 Resumo da Arquitetura Lambda

A Arquitetura Lambda completa é representada na Figura 3 e seus componentes são:

- (A): Todos os novos dados são enviados tanto para a camada *batch* quanto para a camada *speed*. Na camada *batch*, os novos dados são acrescentados ao conjunto de dados mestre. Na camada *speed*, os novos dados são consumidos para fazer atualizações incrementais das visões de tempo real.

- (B): O conjunto de dados mestre é imutável, somente anexável. O conjunto de dados mestre contém apenas a informação mais bruta que não é derivada de qualquer outra informação.

- (C): A camada *batch* pré-computa funções de consulta a partir do zero. Os resultados da camada *batch* são chamados "visões *batch*". A camada *batch* é executada em um laço *while* (*true*) e continuamente re-computa as visões *batch*. A força da camada *batch* é a sua capacidade para computar funções arbitrárias em dados arbitrários. Isto dá-lhe o poder de suportar qualquer aplicação.

- (D): A camada de serviço indexa as visões *batch* produzidas pela camada *batch* e faz com que seja possível obter valores específicos a partir de uma visão *batch* muito rapidamente. A camada de serviço é uma base de dados escalável que faz a troca para novas visões *batch* assim que elas são disponibilizados. Devido à latência da camada *batch*, os resultados disponíveis a partir da camada de serviço estão sempre desatualizados por algumas horas.

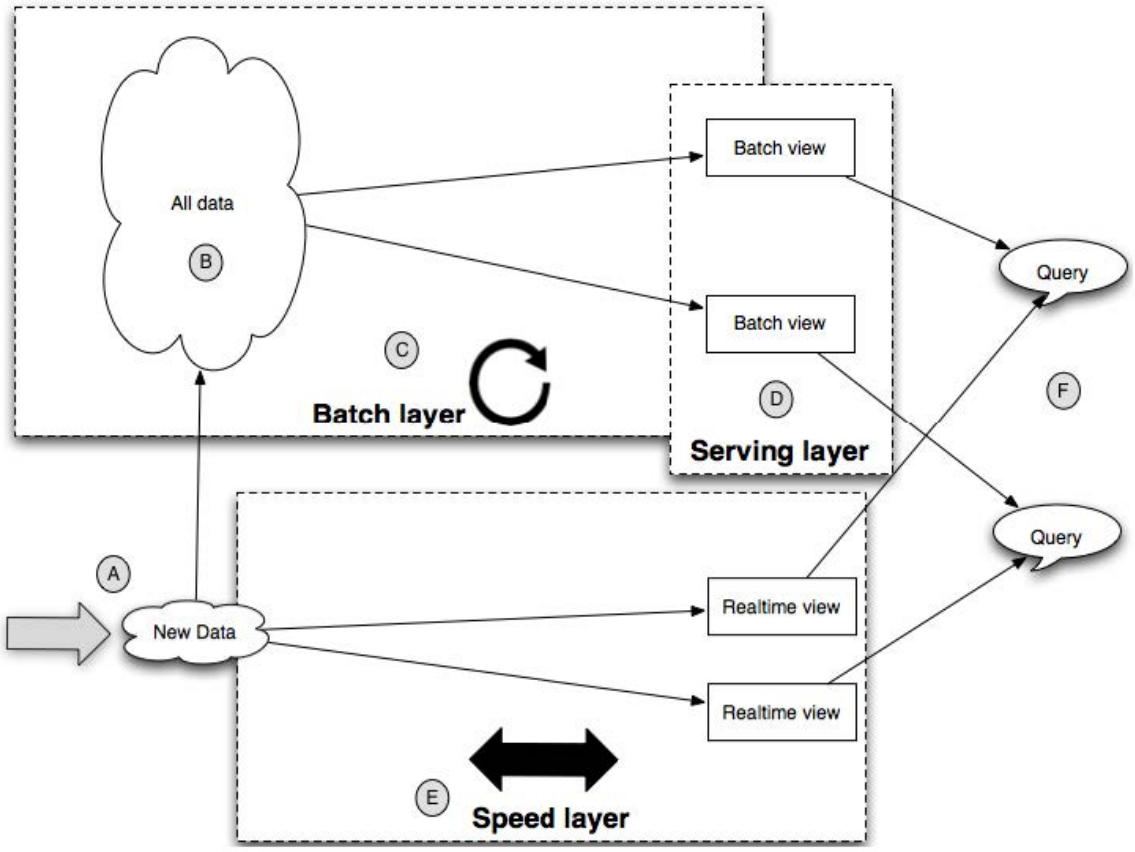

Figura 3 – Diagrama da Arquitetura Lambda. Fonte: (MARZ, 2013)

- (E): A camada *speed* compensa a alta latência das atualizações à camada de serviço. Ela usa algoritmos incrementais rápidos e bases de dados de leitura/escrita para produzir visões em tempo real que estão sempre atualizadas. A camada *speed* só lida com dados recentes, porque todos os dados mais antigos foram absorvidos pela camada *batch* e contabilizados na camada de serviço. A camada *speed* é significativamente mais complexa do que as camadas *batch* e de serviço, mas a complexidade é compensada pelo fato de que as visões de tempo real podem ser continuamente descartadas. Assim, o potencial impacto negativo da complexidade é limitado.
- (F): As consultas são resolvidas pela obtenção de resultados de ambas as visões *batch* e de tempo real e mesclando-os.

2.6 CONSIDERAÇÕES

Nesse capítulo foram apresentados os conceitos e teorias relevantes para fundamentação do contexto ao qual esse trabalho está inserido. Essa fundamentação teórica foi necessária

para a compreensão dos fatores que irão influenciar na definições e desenvolvimento do projeto. Os conceitos foram classificados e discutidos nos seguintes eixos principais:

- **Engenharia Orientada a Modelos (MDE)**. Um estudo sobre a abordagem de MDE se fez necessário para adquirir conhecimento sobre as teorias e práticas adotadas neste contexto para definição do meta-modelos, linguagens de modelagem, Ferramentas de Modelagem e transformações entre modelos. Os conhecimentos adquiridos sobre este tópico são indispensáveis para compreensão das definições da nova abordagem que será proposta nesse trabalho bem como para a implementação desta.

- **Dados Semi-estruturados**. As bases de dados de genoma são publicadas sob formatos semi-estruturados, sendo XML o formato mais adotado pelos publicadores (fontes de dados). Assim, foi necessário investigar os conceitos norteadores desse paradigma de persistência e publicação de dados, principalmente no que diz respeito ao gerenciamento e identificação de seus esquemas. Também foram estudado e discutido o formato de dado semi-estruturados JSON, indispensável para definição de soluções no contexto atual de dados semi-estruturados.

- **Bases de Dados NoSQL**. Os recentes paradigmas de bases de dados NoSQL também foram analisados por apresentarem modelos de dados com definições estruturais e características comuns a dados semi-estruturados. Estes modelos de dados foram classificados como orientado a agregado, que é um tipo de modelo que enfatiza a estrutura aninhada de objetos. No modelo orientado a agregados, as entidades têm propriedades cujo valor também podem ser conjuntos de propriedades e até entidades completas (uma estrutura semelhante a árvore de dados semi-estruturados). Assim, foram apresentados os principais conceitos e modelos das bases NoSQL também necessárias para definição de soluções no contexto atual de dados semi-estruturados.

- **Análise de Dados Biológicos Moleculares**. Os conjuntos de dados sob investigação nesse trabalho estão presentes no contexto de análise de dados biológicos para realização de exames genéticos. Diante disso, se fez necessário estudar os conceitos, rotinas e práticas de todo o fluxo de trabalho para realização de um exame genético. O conhecimento adquirido permitiu compreender as particularidades do ambiente de análise de dados biológicos Moleculares e guiou a definição da abordagem que será proposta nesse trabalho.

- **Arquitetura Lambda**. O sequenciamento do genoma é uma aplicação pioneira do BigData por lidar com dados massivos, variados e de estrutura complexa com dificuldades de armazenamento, análises e visualização. Lambda é uma arquitetura genérica projetada para lidar com dados no contexto de BigData. Como esse trabalho não tem o objetivo de desenvolver uma solução para processamento de dados em

tempo real, mesmo os dados e aplicações tratados no contexto dessa pesquisa serem pioneiros em Big Data, as principais definições e camadas da Lambda serão aplicadas para definição de uma arquitetura de referência para análise de genoma. Para esta, uma nova camada contendo as soluções propostas nesse trabalho será adicionada a arquitetura Lambda. Assim, conhecer as definições da Lambda se faz necessário para a compreensão da arquitetura proposta nesse trabalho.

No Capítulo 3 será apresentado o estado da arte e os trabalhos relacionados a abordagem proposta nessa pesquisa.

3 TRABALHOS RELACIONADOS

"If I have seen further it is by standing on the shoulders of Giants."

—Isaac Newton

Nesse capítulo serão apresentados os trabalhos relevantes que se caracterizam como relacionados a esse trabalho ou que discutem o estado da arte das tecnologias, ferramentas e práticas aplicáveis ao domínio desta pesquisa. Um levantamento bibliográfico foi realizado para identificação destes trabalhos. Após este levantamento, estes trabalhos foram categorizados por suas áreas de concentração e divididos de acordo com a influência e proximidade com a abordagem proposta nesse trabalho. Assim, os trabalhos que representam o estado da arte dessa pesquisa serão apresentados na Seção 3.1 e os trabalhos relacionados serão apresentados na Seção 3.2. Entende-se por estado da arte os trabalhos que abordam ferramentas, tecnologias ou práticas que inspiraram a definição da abordagem proposta nessa tese mas que não estão diretamente relacionados ao domínio sob investigação. Entende-se por trabalhos relacionados os estudos que apresentam contribuições para o domínio de análise de dados biológicos moleculares, especificamente no gerenciamento de bases de dados de genoma.

A Tabela 4 apresenta um guia metodológico resumindo as atividades executadas para identificação das evidências que serão discutidas nesse capítulo.

Etapa de Caracterização do Ambiente - Trabalhos Relacionados	
Objetivo	Conhecimento do estado da arte e identificação dos trabalhos que podem influenciar na definição da abordagem proposta nessa pesquisa
Atividades	Realizar um levantamento bibliográfico e analisar os principais trabalhos com contribuições para o estado da arte e domínio investigado.
Evidências	Categorização das abordagens existentes; Identificação de lacunas; Discussão sobre os trabalhos relacionados.

Tabela 4 – Guia metodológico para a etapa de Caracterização do Ambiente (Trabalhos Relacionados)

3.1 ESTADO DA ARTE

A seguir serão apresentados alguns dos trabalhos estudados classificados nos eixos: Gerenciamento de esquemas; Identificação de esquemas em dados semi-estruturados; e Ferramentas e abordagens para análise de dados de genoma.

3.1.1 Gerenciamento de Esquemas

Há muitos trabalhos acadêmicos que tratam de problemas de gerenciamento de esquemas no contexto de banco de dados (BERNSTEIN; MELNIK, 2007; ATZENI; CAPPELLARI; GIANFORME, 2007; ATZENI et al., 2012; ATZENI; BUGIOTTI; ROSSI, 2014; FAGIN; KOLAITIS; POPA, 2005; HAAS et al., 2005; MILLER; HAAS; HERNáNDEZ, 2000; VELEGRAKIS; MILLER; POPA, 2003) que foram apresentadas desde a formulação original do problema. Contudo, estas abordagens não são suportadas por técnicas de MDE (meta-modelagem).

Em (BERNSTEIN et al., 2000) Bernstein et al. apresenta a possibilidade de uma abordagem de metadados genérica para gerenciamento de esquemas: as suas formalizações teóricas e estudos posteriores convergiram em Rondo, uma plataforma de programação para gerenciamento de esquemas (MELNIK, 2004).

Ainda, *Model Independent Schema and Data Translation* (MIDST) (ATZENI; CAPPELLARI; GIANFORME, 2007) usa um dicionário de modelos e esquemas para representar modelos conceituais e permite transformações transparentes sobre eles mas também não proporciona os benefícios do uso de meta-modelos no contexto de MDE.

Clio (HAAS et al., 2005) visa construir um mapeamento completamente definido entre dois esquemas, dado um conjunto de correspondências definidas pelo usuário e os mapeamentos necessários devem ser definidos manualmente; Além disso, não há nenhum tipo de "*model awareness*", ou consciência de modelo no Clio, que opera em um modelo relacional aninhado generalizado.

A abordagem *Save Our Systems* (SOS) (ATZENI; BUGIOTTI; ROSSI, 2014) suporta a heterogeneidade de três modelos de dados NoSQL (famílias de colunas - Hbase, documento - MongoDB, chave valor - Redis), definindo uma interface de programação comum (que implementa os métodos *put*/*get* e *delete*) que se baseia em um modelo de dados geral. A idéia de um modelo de dados comum teve sua base nas ferramentas MIDST e *Model Independent Schema and Data Translation - Run Time* (MIDSTRT) (ATZENI; CAPPELLARI; GIANFORME, 2007; ATZENI et al., 2012). MIDSTRT é uma abordagem para tradução de modelos e esquemas em tempo de execução baseada na abordagem MIDST.

O trabalho também possui analogias com o framework proposto por (WANG et al., 2015) mas este apresenta solução apenas para gerenciamento de esquemas de repositórios de documentos, especificamente JSON e não faz uso de técnicas de MDE para representar formalmente os esquemas recuperados.

Lenses (YANG et al., 2015) é um framework para curadoria sobre demanda aplicada para dados não-relacionais como objetos JSON ou tabelas web. Além dos demais objetivos deste framework, Lenses apresenta uma abordagem para correspondência entre esquemas (*schema matching*) criando um mapeamento dos esquemas do dados de origem para um esquema de destino definido pelo usuário.

A abordagem proposta por Chillón (CHILLÓN et al., 2017), apresentou uma estratégia para visualização de esquemas inferidos de bancos de dados orientados a agregados para

os principais tipos: documento, chave-valor e família de colunas. A proposta foi validada inicialmente para bases NoSQL orientadas a documento. Esta abordagem se aproxima à pesquisa dessa tese pois também faz uso de uma representação de esquemas baseada em meta-modelos embora estes tenham sua tarefa de identificação de esquemas voltadas apenas para bases NoSQL orientadas a agregados e não para identificação de esquemas em dados semi-estruturados XML.

Embora estes trabalhos possam ser utilizados para extrair, classificar e/ou agrupar uma quantidade considerável de modelos de dados, uma representação formal baseada em meta-modelagem se fez necessária para o domínio sob estudo. A abordagem proposta para representação de bases de dados biológicos moleculares será apresentada na Seção 4.2 do Capítulo 4. E as vantagens de uso das técnicas adotadas serão melhor discutidas no Capítulo 7.

3.1.2 Identificação de esquemas em dados semi-estruturados

Embora uma das principais atrações dos dados semi-estruturados ser o fato de este não exigir a definição explícita de um esquema rígido, vários benefícios da identificação da estrutura de dados foram identificados como: otimizar a avaliação de consultas, facilitar a integração de dados, melhorar o armazenamento, construir índices ou descrever o conteúdo do banco de dados para os usuários (ABITEBOUL; BUNEMAN; SUCIU, 2000). Então, a principal preocupação é definir formalmente o conceito de tipo de dados (tidos como esquema) para dados semi-estruturados. Para estes, os esquemas podem ser definidos por meio de algum tipo de estrutura semelhante a um grafo como grafos rotulados (BUNEMAN, 1997) baseados no Modelo de Troca de Objetos, em inglês *Object Exchange Model* (OEM) (ABITEBOUL, 1996).

Lógica de primeira ordem (*Datalog*) e simulação foram formalismos usados para descrever formalmente a noção de esquema (ABITEBOUL; BUNEMAN; SUCIU, 2000). A identificação de esquema foi principalmente abordada como o problema de encontrar o esquema mais específico ou o tipo aproximado para um conjunto de dados da mesma entidade. Um algoritmo baseado em clusterização para tipificação aproximada de dados semi-estruturados foi apresentado em (NESTOROV; ABITEBOUL; MOTWANI, 1998). Em (WANG; LIU, 1997), um algoritmo para encontrar o tipo típico da maioria dos objetos em uma coleção é descrito. Estes autores testaram o algoritmo no IMDb, em particular escolheram 100 filmes da lista dos 250 filmes mais votados. Em seguida, eles converteram os documentos HTML desses filmes em modelos OEM para executar seu algoritmo e obtiveram três esquemas de objeto mais frequentes, que foram chamados de padrões de esquema.

Um algoritmo para identificar um tipo sucinto (ou seja, um esquema) para um conjunto de dados JSON é proposto em (COLAZZO; GHELLI; SARTIANI, 2012), este funciona em duas fases. Na primeira fase, uma operação *Map-Reduce* é aplicada. A operação *Map* identifica o tipo de cada objeto JSON e gera um par cuja chave é o tipo inferido e o valor é 1. A

operação *Reduce* conta o número de objetos de cada tipo, ou seja, gera um conjunto de pares $<T_i : m_i>$, onde T_i denota o tipo que descreve o conjunto de dados e m_i conta o número de objetos do tipo T_i. Na segunda fase, um algoritmo de fusão de tipos é aplicado para colapsar tipos similares, que é baseado em algumas heurísticas que dependem do valor m_i.

Extração de esquemas XML - O *eXtensible Markup Language* (XML) tem sido o formato predominante para troca de dados na Web. Como a definição de um esquema (por exemplo, um esquema *Document Type Definition*[DTD] ou *XML Schema Definition* [XSD]) não é obrigatória para criar documentos XML, a extração de esquema XML recebeu uma grande atenção da comunidade de banco de dados. O processo de extrair um esquema XML (geralmente um DTD) consiste em duas etapas principais: descobrir a estrutura hierárquica dos documentos e transformá-la na representação do esquema. Em (MOH; LIM; NG, 2000), os autores apresentam uma ferramenta capaz de gerar o DTD de um conjunto de documentos XML. A abordagem aplicada para construir essa ferramenta, chamada de *dtd-miner2000*, funciona da seguinte maneira. Cada documento XML é primeiramente representado na forma de árvore n-ária, então a estrutura geral de todas as árvores de documentos estruturalmente semelhantes é representada em um grafo de abrangência e, finalmente, algumas heurísticas são aplicadas para gerar um DTD a partir das informações no grafo de abrangência.

O Grafo de Identificação de Estrutura (conhecido SG, *Structure Graph*) utilizado na abordagem de Klettke et al. descrito acima é baseado nesta estrutura de dados de grafo abrangente. Em (BEX et al., 2010), o problema de identificar um esquema XML é considerado um problema que "basicamente reduz a aprendizagem de expressões regulares a partir de exemplos positivos de strings", e vários algoritmos são descritos em detalhes. O algoritmo apresentado em (HEGEWALD; NAUMANN; HERSCHEL, 2006) é baseado nas ideias expostas em (MIN; AHN; CHUNG, 2003), onde a forma das expressões regulares é restrita e algumas heurísticas são propostas. Em (JANGA; DAVIS, 2013) uma estratégia para identificar esquemas de bancos de dados XML heterogêneos é apresentada. O esquema é fornecido como uma gramática livre de contexto estendida do Esquema, e as diferentes versões são integradas em uma única gramática que é mapeada para um esquema de banco de dados relacional.

3.1.3 Ferramentas e abordagens para análise de dados de genoma

Muitos sistemas e ferramentas vêm sendo desenvolvidos para suporte à análise de dados de genoma, tanto na indústria como na academia. As ferramentas desenvolvidas suportam partes específicas do fluxo de trabalho de análise de variantes NGS (*Next-generation Genome Sequencing* (PABINGER et al., 2014) mas a etapa de anotação de variantes representa desafios significativos com a necessidade de acesso a múltiplas bases heterogêneas. Embora esta abordagem tenha identificado necessidades em comum com algumas das

abordagens estudadas, a maioria dos trabalhos acadêmicos encontrados tem mantido seu foco na integração dos dados por meio da definição de um esquema global.

Na maioria das ferramentas comerciais disponíveis, os usuários dependem da disponibilidade e do desempenho dos serviços prestados (*on-line*). Outra questão é que muitas dessas ferramentas *on-line* não suportam a submissão por lotes (*batch*) de um conjunto variantes, tornando-as viáveis apenas para a análise manual de um pequeno conjunto de dados. Além disso, problemas legais podem surgir, uma vez que a maioria dos serviços não garante a confidencialidade dos dados. Por outro lado, as ferramentas *off-line* geralmente oferecem mais flexibilidade e não dependem da disponibilidade de qualquer serviço web específico. Ainda, tanto as abordagens *online* como a *off-line* possuem um alto custo para aquisição de licenças (PABINGER et al., 2014).

No que se refere a produções acadêmicas, a maioria das abordagens propõe soluções voltadas para a integração das fontes de dados baseados na definição de um esquema global (BALKO et al., 2004; LIU; LIU; YANG, 2010; LIU et al., 2009; MASSEROLI; CANAKOGLU; CERI, 2016). Entre as técnicas utilizadas nestas abordagens pode-se citar federação de bases de dados, mediadores, consultas a múltiplos bancos dados, armazém de dados, *web services* e *middleware*. Esse trabalho analisou estas abordagens com o intuito de identificar as técnicas e padrões utilizados para representação dos dados, uma vez que a definição de um esquema global não é o o objetivo desse trabalho. Também foram analisadas as técnicas utilizadas para implementação das soluções propostas.

Balko et. al apresentam um serviço chamado *BioDataServer* baseado em mediador (BALKO et al., 2004). Este é apresentado como um serviço de integração, armazenamento, análise e consulta adaptável pelo usuário para dados biológicos moleculares direcionados a clientes comerciais. Pode ser considerado como uma abordagem baseada em mediador que acopla fortemente as fontes de dados participantes em um chamado estoque de dados integrado completamente materializado. Além disso, suportam tipos atômicos (string, inteiros, números) e dados parcialmente sequenciais (através do acoplamento BLAST). Os usuários podem modificar e consultar os dados integrados sem carga de rede. Nesta abordagem, o esquema integrado mescla as informações estruturais das fontes de dados remotas em um esquema semanticamente integrado que anula os limites dos esquemas remotos até aqui separados. O BioDataServer compreende três níveis de esquema. O esquema de exportação remota (RES) que reside nas fontes de dados remotas e fornece os esquemas de dados especificados em seus respectivos modelos de dados nativos. Em outras palavras, o RES é uma descrição estrutural de uma fonte de dados. O esquema do adaptador (AS) que é uma exibição do RES de uma fonte de dados fornecida no modelo de dados canônico do BioDataServer. Por fim, o esquema de usuário integrado (IUS) que reflete uma integração de partes dos esquemas de adaptadores das fontes de dados participantes. O IUS também segue o modelo de dados canônico.

Liu et. al propõem uma solução para integração dos dados baseada em *web services*

(LIU; LIU; YANG, 2010; LIU et al., 2009). Esta abordagem propõe basicamente a criação de um modelo global de banco de dados relacional para criação de uma base pública para compartilhamento dos dados integrados. Este banco foi definido com as informações estruturais das bases consideradas. Estas informações (ou atributos) são mapeadas para entidades (relações ou tabelas) de um modelo global. Os dados são extraídos das fontes heterogêneas em um documento XML que é enviado para o servidor de integração de dados via Web Service. Este é responsável por mapear o documento XML para o banco de dados de público compartilhamento. Este mapeamento faz a relação entre os componentes estruturais das fontes e as tabelas do banco relacional unificado. Esta abordagem se inspirou na taxomonia da *Gene Ontology* para criar o modelo global que propõe a representação de dados específicos inerentes às sequências de DNA. O trabalho não descreve como os demais metadados (que apresentam nomenclaturas e estruturas variadas) são mapeados para o banco de dados de destino.

Masseroli et. al propôs uma arquitetura de software para criar e manter um *data warehouse* integrado, atualizado e publicamente disponível de anotações genômicas e proteômicas (MASSEROLI; CANAKOGLU; CERI, 2016). Esta abordagem também adota um esquema global (relacional) modular e multinível para o gerenciamento integrado dos dados. A característica modular e multinível se dá pelo fato de a camada de importação ser composta de sub-esquemas separados, cada esquema para uma determinada fonte de dados que são individualmente estruturados como na fonte de dados original, ou seja, em uma forma de integração de dados *global-as-view* (XU; EMBLEY, 2004). A integração de uma fonte de dados adicional requer a adição de um sub-esquema para a nova fonte no módulo do recurso cujos dados são fornecidos pela nova fonte, sem afetar outras partes do esquema global. O tratamento modular de cada fonte de dados com seus respectivos esquemas apresenta certa similaridade com a proposta desse trabalho mas, como citado anteriormente, esse trabalho não tem o objetivo de integrar formalmente os esquemas das diversas fontes em um esquema global.

3.2 TRABALHOS RELACIONADOS

Nessa seção, serão discutidos os trabalhos acadêmicos que mais se aproximam da temática dessa pesquisa em seu principal eixo: suporte à anotação de variantes em dados biológicos moleculares (genoma), especificamente na representação das bases de dados. Embora os trabalhos estudados possuam necessidades em comum com a abordagem que será proposta nesse trabalho, observou-se que algumas lacunas ainda precisam ser atendidas.

Na proposta apresentada por (MASSEROLI; CANAKOGLU; CERI, 2016), o tratamento modular de cada fonte de dados com seus respectivos esquemas apresenta certa similaridade com a proposta dessa tese mas, como citado anteriormente, esse trabalho não tem o objetivo de definir um esquema global que integre os diversos esquemas de cada fonte

de dado. Esse trabalho analisou também as abordagens propostas em (LIU; LIU; YANG, 2010; LIU et al., 2009; BALKO et al., 2004) com o intuito de identificar as técnicas e padrões utilizados para representação dos dados e estas também apresentaram definições de um esquema global como solução.

Em relação às estratégias de implementação, as abordagens estudadas pautam suas soluções em Web Services (LIU; LIU; YANG, 2010; LIU et al., 2009), Middlewares (BALKO et al., 2004) e Data Wharehouses (MASSEROLI; CANAKOGLU; CERI, 2016) para gerenciar o esquema global definido. Em todas as abordagens estudadas este esquema global é utilizado para a criação de um banco de dados relacional onde a integração materializada dos dados é realizada.

Um requisito importante para o domínio sob estudo é conseguir manter o histórico versionado (versionamento dos esquemas) e a rastreabilidade dos dados. As abordagens estudadas não apresentam estratégias claras para atender a este requisito como, por exemplo, possibilitar a identificação das diversas versões de esquemas para entidades de mesmo tipo. Com este histórico versionado será possível para um laboratório de genômica reanalisar casos clínicos anteriores que, no momento da primeira análise, poderiam não ter sido solucionados. Essa reanálise pode resultar na reclassificação (nova anotação) da variante, e uma nova notificação pode ser feita ao paciente.

Diante destas evidências, é possível notar que, mesmo com tantos esforços para integrar estas bases, soluções para representação das múltiplas fontes que publicam bases de dados de genoma ainda se fazem necessárias e configuram uma tarefa desafiadora na bioinformática. Definir este tipo de solução é uma tarefa desafiadora na bioinformática pois não existe um padrão de representação para os dados ou resultados de consultas.

Diferentemente das abordagens existentes, a proposta dessa tese não pretende definir um esquema global explícito para integração de dados em bases relacionais. Esta proposta busca aproveitar os benefícios fornecidos por sistemas sem esquemas rígidos explicitamente definidos e adiciona a estratégia de versionamento dos esquemas das fontes e entidades. Nesse trabalho, não há interesse em obter apenas um esquema aproximado para uma entidade, mas sim identificar todas as variações das entidades de mesmo tipo pois, para o domínio dos dados, é importante manter o versionamento e rastreabilidade dos dados. Com o intuito de apresentar um sumário de esquema, a junção dos esquema da entidade é obtido.

Essas características facilitam o acompanhamento da evolução dos dados e o uso de dados não uniformizados, auxiliando o trabalho dos desenvolvedores de aplicativos e de banco de dados para este contexto. A maioria dos bancos de dados NoSQL não exigem a definição explícita de esquemas e são recomendadas para adoção nestes contextos, por motivos como: (1) armazenam dados cuja estrutura está mudando rapidamente ou é tomada como desconhecida, e (2) possuem suporte ao desenvolvimento ágil como um dos seus principais requisitos.

Ainda, as abordagens aqui apresentadas não pautam suas soluções em técnicas de MDE (especialmente meta-modelo, transformações de modelos e linguagens de modelagem) e, portanto, não alcançam os benefícios que estas abordagens podem proporcionar, como permitir a geração de modelos (diagramas) e esquemas de bancos de dados. A meta-modelagem é útil para construir representações em alto nível de abstração da informação que está sendo gerenciada, e as transformações de modelo auxiliam na geração de diagramas e esquemas de bancos de dados. A definição de linguagens de modelagem com notações visuais permitem a criação de ferramentas para visualização e edição dos esquemas em formas de diagramas. Estas técnicas proporcionam o aumento da produtividade, reuso, qualidade, interoperabilidade e manutenibilidade dos sistemas.

3.2.1 Características Observadas e Lacunas Identificadas

A Tabela 5 apresenta um resumo das principais características sobre os trabalhos relacionados bem como apresenta também algumas lacunas identificadas. As principais características analisadas foram:

- Representação dos esquemas - para esta característica buscou-se identificar as estratégias utilizadas pelas abordagens estudadas para representação dos esquemas das bases. Todas os trabalhos estudados adotaram a representação por meio de um esquema global.

- Modelos de Dados - Para esta característica buscou-se identificar os modelos de dados suportados pelas abordagens estudadas. Todas elas adotaram o modelo de dados relacional para implementação do esquema global.

- Evolução dos esquemas - Para esta característica buscou-se identificar se as abordagens estudadas permitem acompanhar a evolução dos esquemas dos dados, mantendo o histórico destes dados e suas estruturadas. Nenhuma das abordagens apresentaram estratégias para prover este recurso.

- Estratégia de Implementação - Para esta característica buscou-se identificar as tecnologias e padrões utilizados na implementação das abordagens estudadas. Estas apresentaram estratégias tecnológicas diversas como *Middleware*, *Web Services* e *Data Warehouse*.

- Ferramentas e Linguagens de Modelagem - Para esta característica buscou-se identificar se as abordagens estudadas apresentaram ferramentas de suporte à diagramação e visualização dos esquemas das fontes de dados. Nenhuma das abordagens apresentou estratégias para visualização destes esquemas.

- Geração de Códigos - Para esta característica buscou-se identificar se as abordagens estudadas permitem a geração de artefatos, ferramentas e esquemas. Nenhuma das

Abordagens	Representação	Modelos	Evolução	Implementação	Modelagem	Geração
(BALKO et al., 2004)	Esquema global	Relacional	–	Middleware	–	–
(LIU et al., 2009)	Esquema global	Relacional	–	Web Services	–	–
(LIU; LIU; YANG, 2010)	Esquema global	Relacional	–	Web Services	–	–
(MASSEROLI; CANAKOGLU; CERI, 2016)	Esquema global	Relacional	–	Data Wharehouse	–	–

Tabela 5 – Resumo dos trabalhos relacionados

abordagens apresentou estratégias que permitam geração de recursos de software ou bancos de dados.

3.3 CONSIDERAÇÕES

Nesse capítulo foi apresentado o estado da arte e os trabalhos relacionados a essa proposta. Com a análise dos trabalhos relacionados evidenciou-se que soluções para representação de bancos de dados biológicos moleculares ainda se fazem necessárias e configuram uma tarefa desafiadora na bioinformática, pois não existe um padrão de representação para os dados e gerenciamento de seus esquemas. Assim, esse trabalho irá propor uma nova abordagem atendendo as lacunas e demais propriedades identificas e discutidas nos trabalhos relacionados. As estratégias consideradas para a definição desta nova abordagem serão discutidas a seguir.

No que diz repeito as estratégias para representação dos esquemas, todos os trabalhos estudados apresentaram abordagens que se baseiam na definição de esquemas globais. Esse trabalho irá propor uma solução que aplicando a abordagem de meta-modelos do contexto de MDE para representar os esquemas das bases de genoma. A meta-modelagem será útil para construir representações em alto nível de abstração da informação que está sendo gerenciada, e as transformações de modelo auxiliarão na geração de diagramas e esquemas de bancos de dados.

sobre os modelos de dados, todos os trabalhos estudados apresentaram abordagens com suporte a apenas o modelo de dados relacional. A abordagem proposta nesse trabalho irá aproveitar os benefícios dos sistemas com esquemas flexíveis e de dados não uniformizados, comuns a bases de dados semi-estruturadas e orientadas a agregados.

Em relação as estratégias para acompanhamento da evolução de esquemas, nenhum dos trabalhos estudados apresentou estratégias para prover este recurso. A abordagem apresentada nesse trabalho irá propor soluções para prover este recurso por meio do uso de esquemas versionados.

Para implementação, os trabalhos estudados apresentaram diversas tecnologias (*Middleware*, *Web Services* e *Data Warehouse*). A proposta desse trabalho se difere das demais

pois fará uso de abordagens do contexto de MDE, especialmente meta-modelso, transformações de modelos, linguagens de modelagem e ferramentas CASE para implementar e validar as soluções propostas.

No que diz respeito a recursos para visualização e diagramação dos esquemas, nenhum os trabalhos analisados apresentou ferramentas e linguagens de modelagem. A solução proposta nesse trabalho irá prover estes recursos por meio da definição de uma linguagem de modelagem e da implementação de uma ferramenta case para criação, visualização e edição de diagramas de esquemas.

Para a geração de recursos de software e banco de dados, nenhum dos trabalhos estudados apresentou estratégias. A solução apresentada nesse trabalho irá propor algoritmos para identificação de esquemas e geração de modelos, e para geração de esquemas de bancos de dados orientados a documento.

As definições da nova abordagem proposta nesse trabalho serão apresentadas no Capítulo 4.

4 UM META-MODELO PARA REPRESENTAÇÃO DE DADOS BIOLÓGICOS MOLECULARES

"The theory of our modern technic shows that nothing is as practical as theory."

—Julius Robert Oppenheimer

Este capítulo apresenta as definições da abordagem proposta nesse trabalho. Esta abordagem foi desenvolvida com técnicas de MDE para representar bases de dados biológicos moleculares (genoma) e permitir o desenvolvimento de ferramentas para suporte à anotação de variantes genéticas. A principal contribuição desta proposta foi o meta-modelo que, além de representar os dados do domínio sob análise, foi aplicado nos processos de identificação de esquemas e criação de ferramentas e bases de dados. Esta implementação permite obter informações sobre a evolução estrutural dos dados e seus esquemas. Assim, a solução aqui apresentada dá suporte à anotação de variantes genéticas, otimizando o entendimento e uso dos dados, impactando diretamente no processo de análise de variantes genéticas para suporte à decisões médicas em diagnósticos. A Tabela 6 apresenta um guia metodológico resumindo as etapas e atividades executadas para identificação das evidências e demais produtos que serão detalhados nesse capítulo.

Na Seção 4.1 serão apresentados os resultados específicos da etapa de análise do domínio descrevendo os conceitos e esquemas identificados para o domínio. Nas Seções 4.2, 4.3 e 4.4 serão apresentados os resultados da etapa de Projeto, com a descrição do meta-modelo,

Etapa de Execução do Processo - Fases: Análise e Projeto	
Objetivo	Analisar e estabelecer o escopo do domínio definindo a arquitetura do domínio e seus componentes.
Atividades	(1) Analisar as bases de dados de genoma identificando as características e conceitos que devem ser representados na solução proposta; (2) Projetar as soluções para representação dos dados e definir algoritmos e demais ferramentas que devem compor a arquitetura da abordagem proposta.
Evidências	Identificação dos conceitos e categorização dos tipos de esquemas; Definição do meta-modelo para representação de dados de genoma; Definição de uma linguagem de modelagem; Definição da arquitetura da plataforma do domínio; Definição de algoritmo de engenharia reversa para identificação de esquemas em bases de genoma; Definição de algoritmo para criação de bases de dados orientadas as documento;

Tabela 6 – Guia metodológico para a etapa de Execução do Processo nas fases de Análise e Projeto

da linguagem de modelagem definida para visualização dos esquemas, e da arquitetura da plataforma do domínio.

4.1 ESQUEMAS VERSIONADOS PARA DADOS BIOLÓGICOS MOLECULARES

As bases de dados investigadas nesta pesquisa relatam informações sobre variações de genoma humano e fenótipos, com evidências para apoio ao processo de análise destas variantes genéticas e suporte às decisões médicas em diagnósticos.

A maioria destas bases é disponibilizada sob formatos de dados semi-estruturados, em geral arquivos XML e VCF (*Variant Call Format*), sejam elas públicas ou privadas. Versões atualizadas destas bases são lançadas periodicamente. Estas versões podem trazer mudanças não só nos dados em si mas também na estrutura (esquema) na qual os dados são disponibilizados. A definição dos conceitos e tipos de esquemas identificados para estas bases de dados biológicos moleculares serão apresentadas a seguir:

- *Source* - Uma fonte de dados é uma base de dados. Estas são identificadas pelos nomes de seus publicadores. *ClinVar* e *GenBank* são exemplos de fontes de dados.
- *SourceVersion* - Uma versão da fonte é uma publicação da base de dados que é lançada periodicamente. A fonte *Clinvar*, por exemplo, publica novas versões de seus dados a cada mês.
- *Entity* - Uma entidade é a representação de uma variante de genoma. *ClinVarSet* é um exemplo de uma entidade. Nas bases de dados analisadas, uma entidade de mesmo tipo pode apresentar variações em sua estrutura. Cada variação será tratada como uma versão da entidade.
- *Versão* - As versões são os conjuntos de objetos de mesmo rótulo de *Entity* ou *SourceVersion*, que têm diferentes esquemas. *ClinVarSet1* é um exemplo de uma versão da entidade *ClinVarSet*.

Como já mencionado, os dados das diversas fontes (*Sources*) são disponibilizados em versões de documentos semi-estruturados (*SourceVersion*) e cada documento V possui, por sua vez, um conjunto de instâncias de variantes E_i (*Entity*)com dados de anotações de genes $V(E_0, E_1, ..., E_n)$.

Cada entidade E possui diversos atributos (*Attributes*) a_i (também conhecidos como propriedades ou campos) $E(a_0, a_1...a_n)$. Cada atributo pode ser especificado como um par (n_i, v_i) onde n e v são nome e valor, respectivamente. O *Valor* de um atributo pode ser:

- Atômico - que pode ser um número, uma cadeira de caracteres (*string*) ou um booleano.

- Outro atributo - podendo ter um conjunto de atributos (n,v) embutidos em um campo ao qual estes pertencem. Esta estrutura possibilita o armazenamento agregado (aninhado) de uma entidade completa dentro do valor de um atributo.

- Array - um conjunto de valores que pode ser homogêneo ou heterogêneo.

A Figura 4 apresenta um exemplo de uma versão da fonte de dados *ClinVar*. Nela pode-se identificar a presença de seis instâncias da entidade *ClinVarSet*. A última instância ilustrada na figura apresenta um conjunto de elementos como, por exemplo, *RecordStatus*, *Title* e *ReferenceClinVarAssertion*, e este último possui diversos atributos embutidos.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ReleaseSet Dated="2018-06-05" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" Type="full" xsi:
noNamespaceSchemaLocation="http://ftp.ncbi.nlm.nih.gov/pub/clinvar/xsd_public/clinvar_public_1.53.xsd">

<ClinVarSet ID="18549596">
</ClinVarSet>

<ClinVarSet ID="18549599">
</ClinVarSet>

<ClinVarSet ID="18549600">
</ClinVarSet>

<ClinVarSet ID="18549601">
</ClinVarSet>

<ClinVarSet ID="18549602">
</ClinVarSet>

<ClinVarSet ID="18549603">
  <RecordStatus>current</RecordStatus>
  <Title>NM_001201551.1(CFHR4):c.-33981_58+5231del AND Gestational diabetes mellitus uncontrolled</Title>
  <ReferenceClinVarAssertion DateCreated="2015-02-25" DateLastUpdated="2017-04-05" ID="381062">
    <ClinVarAccession Acc="RCV000161191" Version="1" Type="RCV" DateUpdated="2017-04-05"/>
    <RecordStatus>current</RecordStatus>
    <ClinicalSignificance>
      <ReviewStatus>no assertion provided</ReviewStatus>
      <Description>not provided</Description>
    </ClinicalSignificance>
    <Assertion Type="variation to disease"/>
    <ObservedIn>
      <Sample>
        <Origin>unknown</Origin>
        <Species TaxonomyId="9606">human</Species>
        <AffectedStatus>yes</AffectedStatus>
      </Sample>
      <Method>
```

Figura 4 – Exemplo de uma versão do ClinVar. As instâncias de entidade são delimitadas pela tag *ClinVarSet*.

Também foi identificado o conceito de *Associação*, podendo ser um atributo de valor inteiro ou *string*, que faz referência a um atributo de outra entidade. A representação exata dessa associação depende da implementação. No caso do domínio em análise essa associação é útil para representar relacionamentos entre as diversas variantes de um mesmo gene. Este recurso não é implementado nas bases de origem (Semi-estruturadas) mas possui potencial para utilização em consultas nas bases para onde os dados serão destinados.

Os esquemas da bases analisadas podem ser definidos por meio de algum tipo de estrutura semelhante a um grafo (como grafos rotulados baseados no Modelo OEM, propostos por Buneman (BUNEMAN, 1997) e Abiteboul (ABITEBOUL, 1996)). A seguir, será detalhado como a estrutura das bases de dados biológicos moleculares se adéquam a estrutura de árvore (um tipo especial de grafo):

- Um nó raiz possui nós filhos para cada entidade que por sua vez possuem nós filhos para cada atributo.
- Nós para valores atômicos ou valores de referências não possuem nós filhos (se tornam nós folhas).
- Nós para outros atributos ou entidades embutidas tem um nó filho para cada atributo e/ou entidade que sucessivamente tem nós filhos para cada atributo.
- Nós para arrays tem nós filhos para cada elemento do array.
- As arestas partindo de um nó entidade possuem os nomes dos atributos e os nós subsequentes possuem os seus valores.
- As arestas partindo de um nó Versão possuem os atributos identificadores das entidades.

O número de níveis ou profundidade de uma árvore é determinado pelo nível mais alto de aninhamento dos objetos(entidades e atributos) incorporados. Cada nó é associado a um caminho (*path*) que resulta da travessia pelas arestas partindo do nó raiz para o nó de destino. Um caminho, então, pode ser uma sequência ordenada de rótulos das arestas percorridas (isto é, nomes de campos e ou entidades) separadas pelo símbolo "." (ponto). A Figura 5 ilustra apenas uma entidade de exemplo para simplificar a visualização. Nela podemos ver como exemplo o *path ClinVarSet.ReferenceClinVarAssertion.Assertion.Type* que apresenta o dado *"variation to disease"*.

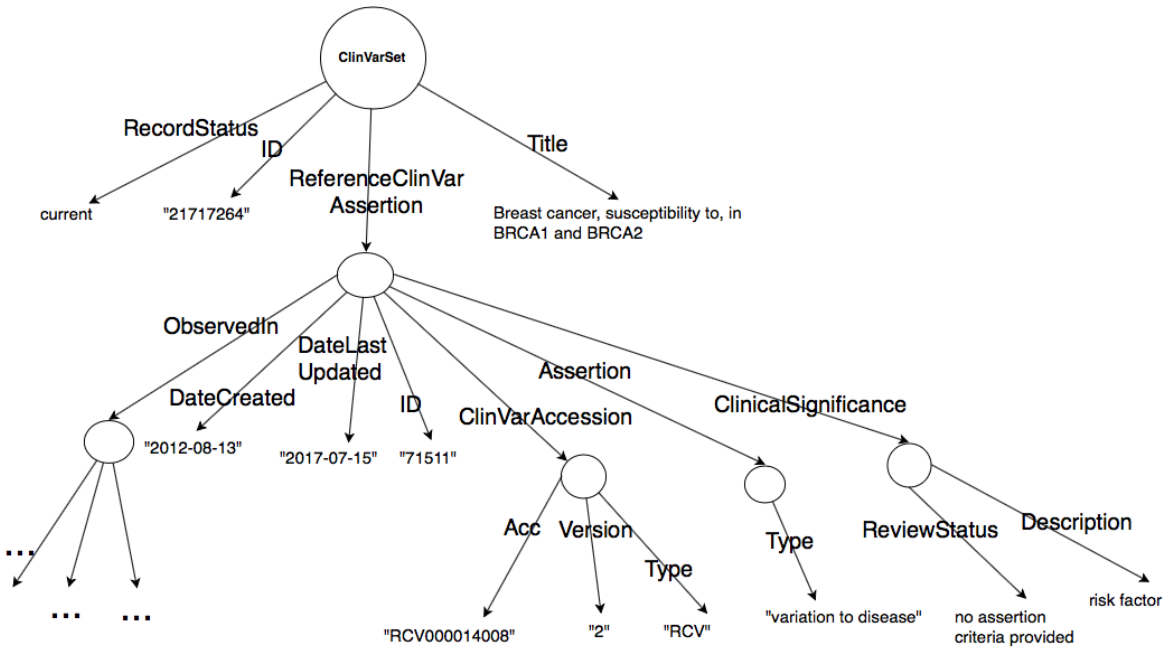

Figura 5 – Exemplo de uma instância *ClinVarSet* ilustrado no formato de árvore.

O esquema das bases pode ser obtido substituindo os valores atômicos dos atributos pelos seus tipos de dados. Portanto, um esquema tem a mesma estrutura que o descrito em relação aos atributos e entidades. Ou seja, um esquema também pode ser representado por uma estrutura de dados em forma de árvore cujos nós folhas são rotulados com identificadores de tipos de dados em vez de valores atômicos, como descrito anteriormente. A Figura 6 ilustra o esquema da instância presente na Figura 5.

O conjunto de tipos primitivos de dados depende da linguagem de representação destes. Este trabalho considerou inicialmente os dados disponibilizados em documentos XML e seus tipos primitivos: Numéricos, cadeia de caracteres e temporais. Tipos array são considerados para objetos embutidos e listas de valores.

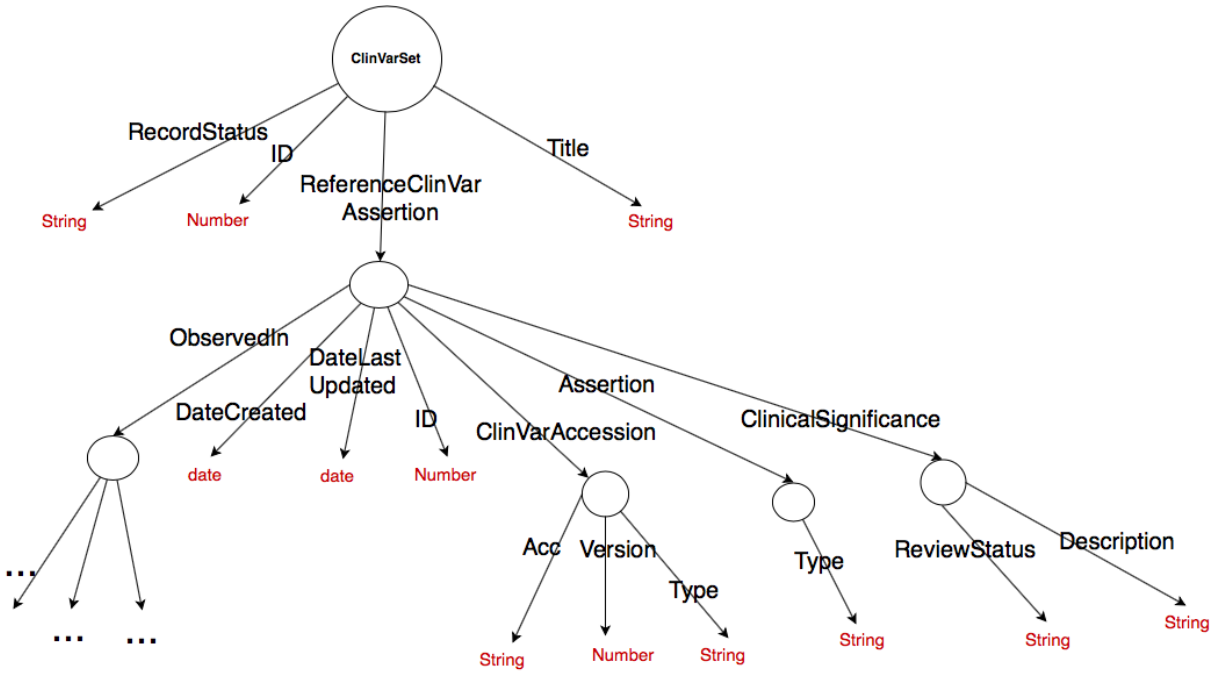

Figura 6 – Exemplo de esquema para a instância *ClinVarSet* em formato de árvore.

No domínio investigado, um banco de dados armazena dados sobre as variantes de genoma (entidades). Uma entidade rotula todos os atributos que se referem para o mesmo conceito (i.e. variante). Como mencionado anteriormente, a natureza sem esquema fixo dos dados semi-estruturados (também comum aos bancos de dados NoSQL) permite que diferentes objetos do mesmo tipo de entidade sejam armazenados com variações em seus esquemas. Nesse trabalho, não há interesse em obter apenas um esquema aproximado para uma entidade, mas sim identificar todas as variações das entidades de mesmo tipo pois, para o domínio dos dados, é importante manter o versionamento e rastreabilidade para acompanhamento da evolução dos dados. Portanto, pôde-se introduzir a noção de versão não só para cada fonte de dados (*SourceVersion*) mas também para suas entidades (*Entity*). As versões denotam os conjuntos de objetos que, compartilhando o mesmo ró-

tulo de *Entidade* ou *SourceVersion*, têm diferentes esquemas. Assim, cada *SourceVersion* pode ter uma ou mais versões de entidades. Com o intuito de apresentar um sumário de esquema, a junção dos esquema da entidade também é obtido.

A existência de versões, cada uma com diferentes níveis de variação em seu esquema, levanta a necessidade de esquemas versionados, que denotam esquemas definidos para o conjunto de versões. Isso é diferente dos bancos de dados tradicionais ou linguagens de programação, onde uma entidade tem apenas um esquema.

Levando em consideração as versões das fontes, um esquema dessa fonte não pode ser apenas um esquema: ele tem que conter o conjunto de esquemas das diferentes versões de suas entidades em particular. Assim, um esquema geral da fonte também pode ser definido como o conjunto dos esquemas de suas diversas versões publicadas.

Em resumo, é possível ter três tipos de esquema no domínio estudado:

1. O Esquema da Versão da Fonte, com as representações das diversas versões de suas entidades;

2. O Esquema da Fonte, com a união dos esquemas de todas as suas versões; e

3. O Esquema da Entidade, com a junção de todos os atributos presentes nas entidades de mesmo tipo.

Este último esquema é útil não só para sumário e visualização dos detalhes de cada tipo de entidade mas também para situações onde pretende-se representar os dados em formatos de bases tradicionais que necessitem de um esquema único para todas as instâncias de entidades de mesmo tipo.

4.2 DEFINIÇÃO DO METAMODELO GenDB - UM META-MODELO PARA REPRESENTAÇÃO DE DADOS BIOLÓGICOS MOLECULARES

Definir um meta-modelo que possa ser empregado para expressar grandes conjuntos de dados biológicos moleculares é um dos desafios dessa pesquisa.

A Figura 7 apresenta o meta-modelo que foi definido de acordo com os conceitos introduzidos na seção anterior. Este meta-modelo será chamado de GenDB.

A representação completa (metaclasse *GenDB*) pode possuir várias fontes de dados (metaclasse *Sources*) que por sua vez podem possuir várias versões (metaclasse *SourceVersion*). Sobre cada versão são capturados o id(*versionId*), a data de criação (*createDate*), e o status que indica se esta versão é um lançamento histórico (*release*) ou a última versão lançada (*latest*). Cada versão pode possuir várias entidades (metaclasse *Entity*). Como

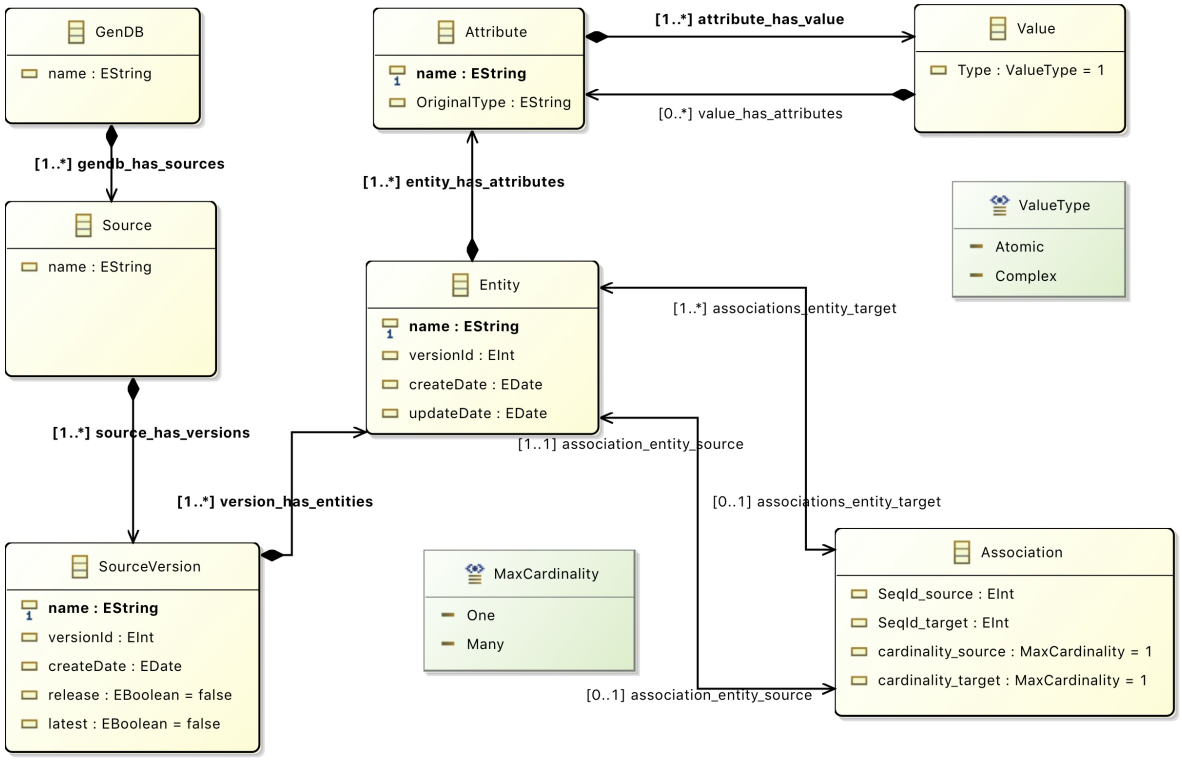

Figura 7 – O Meta-modelo GenDB para representação de dados biológicos moleculares.

mencionado na Seção anterior uma entidade representa as variantes (que pode ser comparada a uma entidade no modelo ER ou uma coleção no modelo orientado a documentos). Sobre as entidades são capturados o nome *Name*, o id da versão da entidade(*versionId*), e as datas de criação (*createDate*) e de atualização (*updateDate*) da entidade. Todas essas informações permitem a rastreabilidade e versionamento dos dados.

A metaclasse *Attribute* representa as propriedades de cada entidade (que pode ser comparada a um atributo no modelo ER). Este atributo pode ser um campo de dados simples ou uma associação. Cada atributo possui, além do par (nome, valor), o tipo de dado original do dado, capturado pelo atributo *OriginalType*. O valor de um atributo (metaclasse *Value*) pode ser um valor atômico ou complexo (i. e., uma lista de valores ou outros atributos aninhados). O tipo do valor é definido pelo tipo de dado ENUM *ValueType* e capturado pelo atributo *Type*. Para um valor atômico, o tipo do valor é indicado com *Type:ValueType=Atomic*, por exemplo. Para um valor complexo o tipo do valor é indicado como *Type:ValueType=Complex*, por exemplo, e a relação *value_has_attributes* indica os atributos que estão aninhados. O valor do tipo complexo permite a representação de sub-atributos e objetos agregados (que podem ser comparados a documentos embutidos em bancos de dados orientados a documento).

Uma associação (metaclasse *Association*) é utilizada para representar referências entre entidades. Esta pode ser comparada a uma chave estrangeira no modelo relacional ou

uma referência a um documento no modelo orientado a documentos. Os relacionamentos *associations_entity_target* e *associations_entity_source* são necessários para tornar a referência bidirecional e permitir a navegação partindo tanto da origem(*source*) quanto do destino (*target*). A cardinalidade desta associação é definida pelo tipo de dado ENUN *MaxCadinality* e capturada pelos atributos *cardinality_source* e *cardinality_target*.

Este meta-modelo foi aplicado inicialmente em um processo que envolve a identificação de esquemas em bases de dados XML e criação de bases de dados orientadas a Documento (*MongoDB*). Essa implementação será apresentada no Capítulo 5.

4.3 DEFINIÇÃO DA LINGUAGEM GenML - UMA LINGUAGEM DE MODELAGEM PARA ESQUEMAS DE DADOS BIOLÓGICOS MOLECULARES

Nesta pesquisa foi definida a linguagem de modelagem GenML e uma ferramenta de modelagem que permitem a visualização em forma de diagramas (modelos) dos esquemas discutidos na Seção 4.1 (o esquema da fonte, o esquema da versão da fonte, e o esquema da entidade). A ferramenta de visualização também permite a edição e criação de modelos utilizando a linguagem de modelagem definida. No contexto de uma linguagem, um modelo define a sintaxe concreta (ou notação gráfica) e o meta-modelo define a sintaxe abstrata (ou gramática) e a semântica (ou interpretação dos conceitos). Na criação da linguagem GenML, primeiramente foi definida a notação visual (sintaxe concreta) para o meta-modelo (sintaxe abstrata). Tomando como entrada o meta-modelo e conhecendo sua notação visual, a ferramenta de modelagem permite criar visões (chamados de diagramas) de modelos que estejam em conformidade com o meta-modelo de entrada.

Como dito anteriormente, foi definida uma linguagem de modelagem com uma notação visual para a representação adequada dos esquemas. A seguir, serão apresentados os diagramas propostos para os esquemas e os princípios adotados na definição da notação visual para criação de modelos.

4.3.1 Diagramas para Esquemas de Dados Biológicos Moleculares

A notação da linguagem de modelagem foi desenvolvida seguindo alguns dos princípios apresentados por Moody (MOODY, 2010) para construção de notações visuais. Entre estes princípios estão: Claridade Semiótica, Discriminação Perceptiva, Transparência Semântica, Gerenciamento da Complexidade, Integração Cognitiva, Codificação Dupla e Economia Gráfica. A seguir serão apresentados detalhes sobre cada um destes princípios:

Claridade Semiótica - O princípio da claridade semiótica define que deve haver uma correspondência de 1:1 (um para um) entre os construtores semânticos (meta-modelo) e os símbolos gráficos da linguagem. A Figura 8 ilustra o mapeamento entre os construtores semânticos e a sintaxe visual.

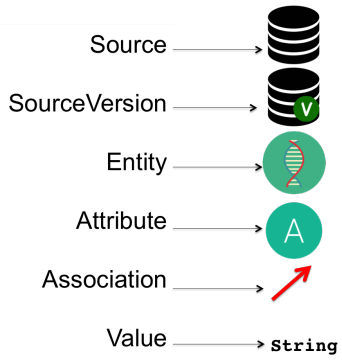

Figura 8 – Mapeamento entre o meta-modelo e os símbolos gráficos da sintaxe visual da GenML

É importante mencionar que o elemento *Value* não possui um símbolo gráfico associado a sua representação visual para limitar a complexidade gráfica e diagramática do modelo. Este componente aparece representado (textualmente) de acordo com seus tipos. No papel de valor atômico, o tipo de dado é associado ao nome do atributo e, no papel de valor composto por outros atributos, a lista dos sub-atributos é apresentada. É possível que atributos contenham valores atômicos associados a ele e ainda assim apresentarem subatributos em sua composição. Com o intuito de melhorar a expressividade dos atributos embutidos, um estilo condicional foi aplicado com uma adaptação do símbolo original de atributo. Para estes o símbolo é marcado com uma seta vermelha (indicando aninhamento) e é apresentado em menor tamanho que os atributos pais. Esta diferenciação do elemento *Value* e dos atributos aninhados está ilustrada na Figura 9.

Discriminação Perceptiva - O princípio da discriminação perceptiva define que os diferentes símbolos devem ser claramente distinguidos um do outro. Além da clara diferenciação visual entre os símbolos gráficos ilustrados na Figura 8, a linguagem faz uso de duas técnicas visuais para demonstrar a hierarquia entre os elementos: (1) uma organização baseada em contêineres, e outra (2) que organiza os elementos do modelo como uma árvore hierárquica. Estes diagramas podem ser vistos nas Figuras 9 e 11, respectivamente. A técnica visual de contêiner foi utilizada também na definição do diagrama de entidade, que permite visualizar horizontalmente os atributos de uma determinada entidade (Figura 10).

Transparência Semântica - O princípio da transparência semântica define que devem ser utilizadas representações visuais em que sua aparência sugira o significado. Na lingua-

gem GenML foram utilizadas figuras clássicas de representação de bases de dados para indicar as fontes de dados e suas versões, sendo este último marcado com um "V", sugerindo que se trata de uma versão da fonte. Para os atributos foi utilizado um símbolo gráfico com um "A", sugerindo o significado para atributo. Para a Entidade foi utilizado um símbolo gráfico com a ilustração de uma sequência de DNA, uma vez que a entidade aqui representada significa uma instância de uma variação de genoma no conjunto de dados a serem modelados. Para o elemento associação foi utilizado uma seta que também sugere o significado de uma referência entre dois elementos.

Gerenciamento da Complexidade - O princípio da gerência de complexidade define que devem ser incluídos mecanismos explícitos para lidar com a complexidade. Como mencionado anteriormente, a linguagem GenML definiu duas técnicas visuais para apresentação dos modelos (contêineres e árvore), além de ter um diagrama específico para visualização do esquema de uma entidade. Estes diagramas foram definidos para reduzir a complexidade da linguagem quando há a necessidade de escalar a visualização dos modelos (muitas entidades por versão da fonte, por exemplo) permitindo uma visualização detalhada de todos os elementos que compõem cada esquema.

As Figuras 9, 10 ilustram exemplos de diagramas de contêineres para os esquemas da fonte e das versões da fonte, e da entidade, respectivamente. A Figura 11 ilustra o diagrama do esquema da fonte em formato de árvore.

Integração Cognitiva - O princípio da integração cognitiva define que devem ser incluídos mecanismos explícitos para suportar a integração de informação entre os diferentes diagramas. Na linguagem GenML todos os diagramas fazem uso dos mesmos elementos e seus respectivos símbolos gráficos. Esses símbolos são igualmente utilizados em todos os diagramas para fornecer uma visão uniforme ao usuário e possibilitar a integração cognitiva da informação.

Expressividade Visual - O princípio da expressividade visual define que uma gama completa de variáveis e capacidades visuais devem ser utilizadas. Como mencionado anteriormente, a linguagem GenML definiu duas técnicas visuais para apresentação dos modelos (contêineres e árvore), além de variados símbolos gráficos para representação individual dos elementos semânticos.

Codificação Dupla - O princípio da codificação dupla define que elementos de texto podem ser utilizados para complementar os elementos gráficos. Nos diagramas construídos com a GenML há associação entre os nomes dos elementos (construtores de um esquema) e suas representações visuais. Como exemplo, a Figura 9 apresenta o nome da fonte *ClinVar* ao lado símbolo gráfico que representa o elemento semântico *Source*.

Economia Gráfica - O princípio da economia gráfica define que o número de diferentes símbolos gráficos devem ser cognitivamente gerenciáveis. Como mencionado anteriormente, o elemento *Value* não possui um símbolo gráfico associado a sua representação visual para limitar a complexidade gráfica e diagramática do modelo. Além disso, os dois tipos de diagramas (contêiner e árvore) foram criados para aumentar o nível de expressividade do modelo.

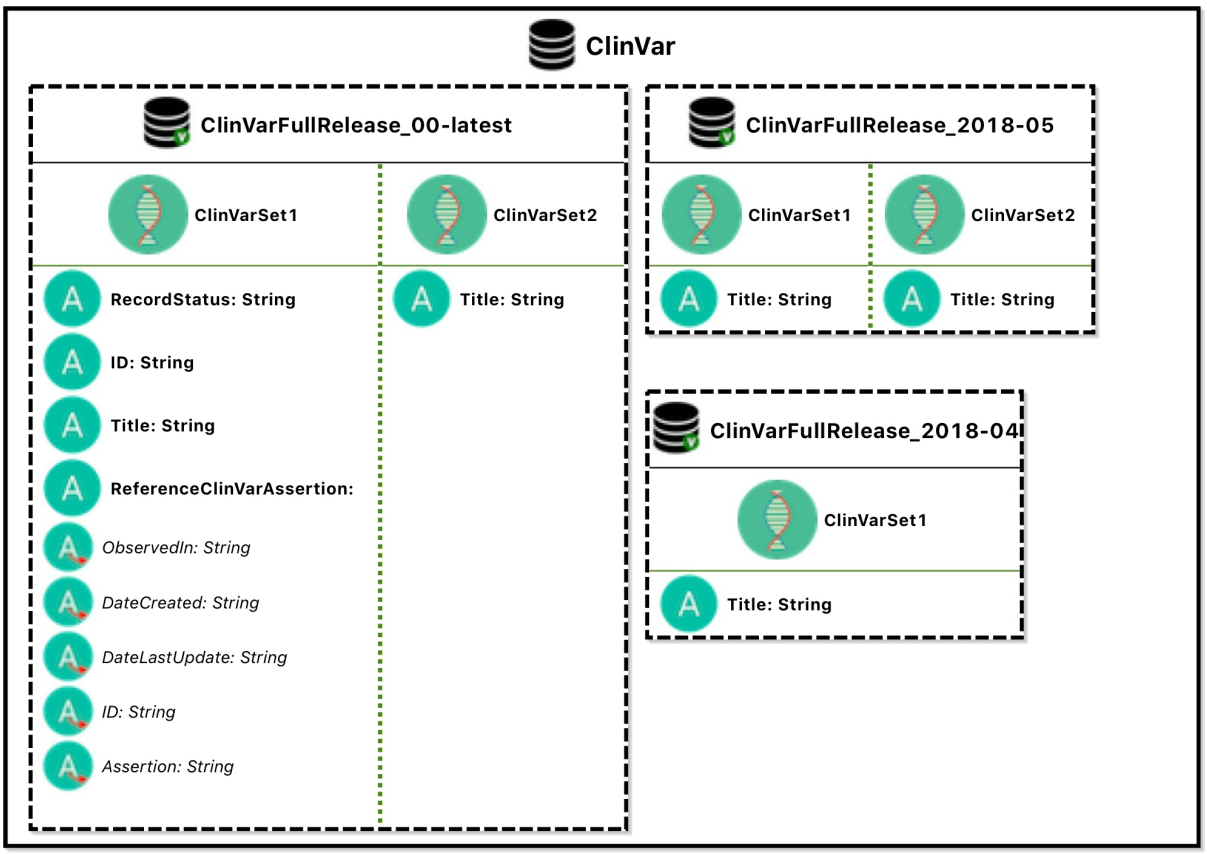

Figura 9 – Diagrama da fonte *ClinVar* com três versões *ClinVarFullRelease_00-latest*, *ClinVarFullRelease_2018-05* e *ClinVarFullRelease_2018-04*

A ferramenta de modelagem utiliza os mesmos símbolos gráficos utilizados na notação da linguagem de modelagem para a caixa de ferramentas do editor, permitindo que os usuários identifiquem os elementos na criação e personalização de esquemas. Esses símbolos gráficos são usados em todos os diagramas criados para fornecer uma visão uniforme ao usuário. Ainda, é possível navegar do diagrama de contêineres para os diagramas de entidade e árvore por meio de um duplo clique nos elementos *Entity*, e *Source*, respectivamente.

É importante mencionar que a ferramenta de modelagem é capaz de realizar a validação dos diagramas, verificando se estes estão em conformidade com o meta-modelo. Esta funcionalidade evita que os usuários façam uso incorreto dos construtores do modelo.

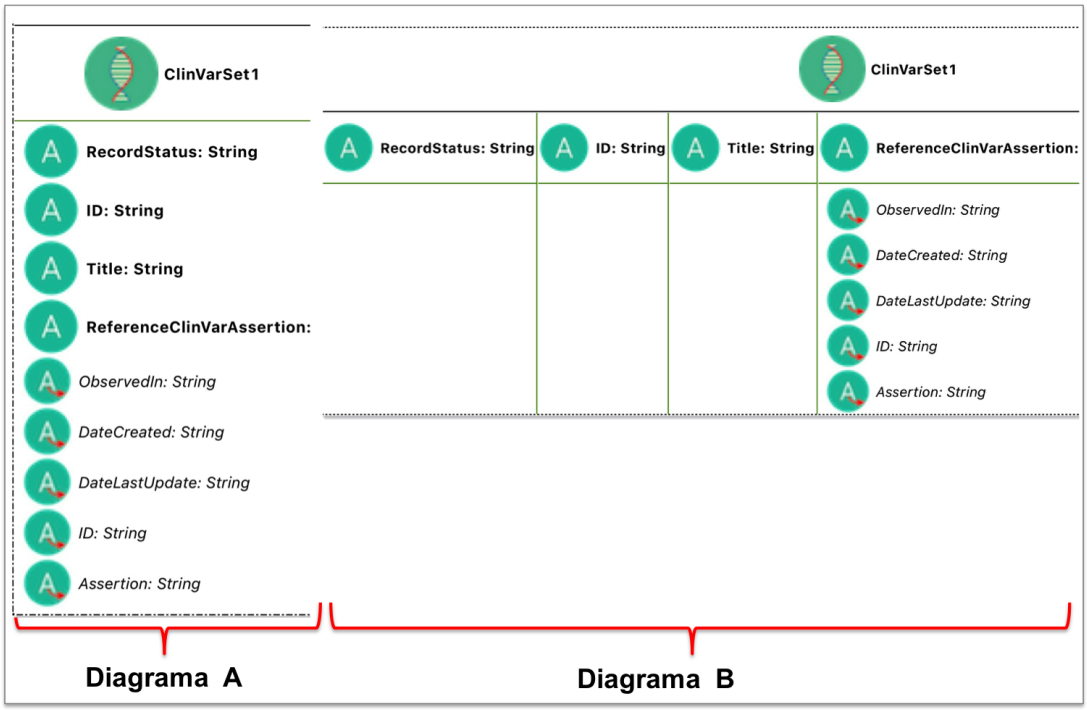

Figura 10 – Diagramas da entidade *ClinVarSet1*

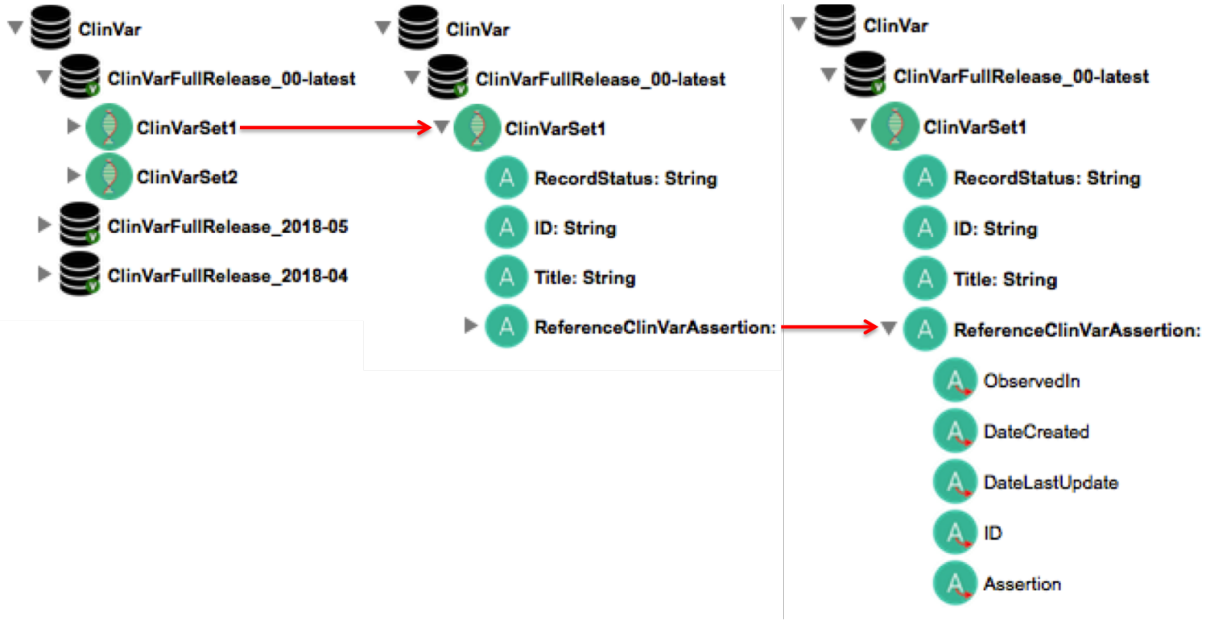

Figura 11 – Diagrama da fonte *ClinVar* em formato de árvore

4.4 DEFINIÇÃO DA ARQUITETURA DA PLATAFORMA DO DOMÍNIO

Primeiramente é importante definir o conceito de plataforma. Nesse trabalho, entende-se por plataforma um conjunto integrado de elementos computacionais que permitem

o desenvolvimento e a execução de uma classe de produtos de software (SILVA, 2015; VOELTER et al., 2013; GREENFIELD; SHORT, 2003; BRAMBILLA; CABOT; WIMMER, 2012).

Esta pesquisa definiu uma plataforma baseada em meta-modelagem para produtos de software relacionados ao processo de análise de dados de genoma humano, especificamente para a etapa de anotação de variantes genéticas.

A plataforma definida utiliza alguns benefícios da Arquitetura Lambda (MARZ, 2013). Como detalhado no Capítulo 2 esta arquitetura contribui com o problema de manipular grandes volumes de dados em tempo real. Esta plataforma possui uma estrutura decomposta em quatro camadas: "*batch layer*", responsável por persistir os dados históricos; "*serving layer*", responsável por realizar análises ou criar visões sobre os dados e disponibilizá-los através das visões definidas; e "*speed layer*", onde são realizadas análises em tempo real. Todas essas camadas podem receber consultas da aplicação final e os dados presentes nelas podem ainda ser computados, cruzados ou agregados.

Nesse trabalho, uma nova camada "*Meta-layer*"foi adicionada à arquitetura Lambda. Esta camada possui os componentes e artefatos de MDE (*SchemaDiscovery*, *meta-modelos* e geradores) necessários para representação e transformação dos dados. Em resumo, esta camada é responsável por adquirir os dados, identificar os esquemas das fontes (*Sources*) e os converter para o padrão adotado na arquitetura do domínio.

Obter os esquemas das fontes de dados envolve identificar as entidades, as versões de cada entidade, e os campos(atributos) e relações que possam existir em cada versão. Portanto, um algoritmo de engenharia reversa deve percorrer todos os objetos armazenados (entidades) e analisar sua estrutura para identificar todos os elementos do esquema, ou seja, identificar o esquema implícito nos dados.

A engenharia reversa pode se beneficiar das técnicas MDE pois meta-modelos fornecem um formalismo para representar o esquema obtido em um alto nível de abstração. Ainda, a geração de códigos é facilitada pelo uso de transformações de modelo. Portanto, criou-se uma solução MDE para realizar a engenharia reversa identificando os esquemas de fontes de dados semi-estruturadas XML (*camada Meta-Layer* da Figura 12).

A Figura 12 ilustra a arquitetura da plataforma com todos os elementos propostos mas este trabalho limitou-se a definir e validar a camada (*Meta-layer*) por ser onde se concentram as principais contribuições desta pesquisa. As demais camadas seguem as especificações técnicas padrões da arquitetura Lambda já descritas no Capítulo 2. Na camada *Meta-layer*, primeiramente os esquemas são identificados nas fontes de dados. O processo de engenharia reversa analisa todos os elementos das fontes (por exemplo, entidades, atributos e versões) e gera um modelo em conformidade com o meta-modelo GenDB. Esse modelo pode ser visualizado em forma de diagramas com o uso de uma ferramenta de modelagem (ferramenta *case*). Por último, os modelos identificados podem ser usados para geração de ferramentas, que exigem conhecimento da estrutura do banco de dados (por exemplo, um mecanismo de consulta SQL, validadores de dados ou diagramas de

esquema, entre outras aplicações). Estas ferramentas ajudam os desenvolvedores a lidar com problemas causados pela ausência de um esquema explícito. Transformações T2M e M2T são usadas para implementar algumas dessas ferramentas e serão melhor descritas nas Seções 4.4.1 e 4.4.2, respectivamente.

É importante ressaltar que a plataforma é extensível permitindo que novos componentes customizados sejam gerados fazendo uso do meta-modelo definido. O uso do meta-modelo GenDB para a representação dos dados garante flexibilidade em relação às fontes de dados que podem ter seus esquemas identificados e visualizados diagramaticamente com a GenML, uma vez que a maioria das fontes de dados disponibilizam suas bases no formato XML. Isto será demostrado no Capítulo 6.

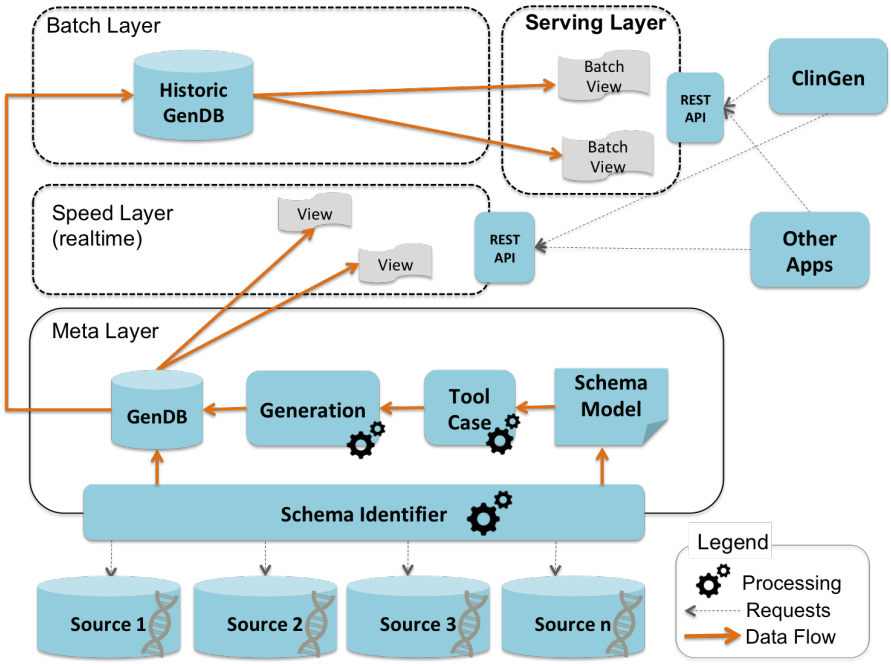

Figura 12 – Arquitetura da plataforma do domínio.

Assim, esta abordagem utiliza a generalidade do meta-modelo para permitir uma prática de desenvolvimento que não está vinculada a uma (*Application Programming Interface*(API) de um Sistema de Gerenciamento de Banco de Dados (SGBD) específico, mas permitindo a atendimento a fontes diversas. Ainda, os elementos que compõem esta plataforma são modulares e independentes aumentando a possibilidade de reuso dos mesmos.

4.4.1 DEFINIÇÃO DO ALGORITMO DE ENGENHARIA REVERSA PARA IDENTIFICAÇÃO DE ESQUEMAS VERSIONADOS

No algoritmo para identificação de esquemas, as fontes de dados (*Sources*) são analisadas para identificação dos elementos que compõem o esquema da base: *SourceVersions*, *Entities* e *Attributes*. Essa análise envolve um processo de engenharia reversa que deve obter um esquema para cada fonte, versão da fonte e entidade. Portanto, o processo de engenharia reversa identifica os elementos de um esquema analisando a coleção de instâncias presentes no arquivo XML da fonte. O resultado é um modelo que está em conformidade com o Meta-modelo GenDB. À medida que os elementos do esquema são identificados, eles são representados como elementos do diagrama (modelo.gendb) gerado como saída. Para isso, o algoritmo é organizado em dois estágios. Primeiro, Entidades, Atributos e Tipos de dados (primitivos e atômicos ou arrays com subatributos agregados) são identificados e, em seguida eles são transformados para os elementos correspondentes no meta-modelo. Neste segundo estágio é que os modelos e seus diagramas são criados. O Algoritmo 4.1 apresenta o comportamento definido para identificação de esquemas versionados.

```
1  Algorithm Identifier
2  Input: base.xml
3  Output: model.gendb
4
5  BEGIN
6      READ base.xml
7      SET Versions as Array to empty
8      SET EntitySchema as Array to empty
9      SET EntityVersion as Array to empty
10     SET SourceSchema as Array to empty
11     SET SourceVersionSchema as Array to empty
12     SET Chield as Array to empty
13     SET SourceName GET Name on the Source
14     SET SourceVersionName GET ReleaseSetName on the Source
15     SET Instances GET Instances on the SourceVersion
16     SET EntityName GET InstanceName on the Instances
17
18     FUNCTION getStructure /*analisa o arquivo XML e identifica a
           estrutura*/
19     SET v to 1
20     FOR each node in Instances TILL final node  /*para cada atributo*/
21         SET Name GET Name on the node
22         SET OriginalType GET datatype on the node
23         IF OriginalType is "ATOMIC"
24             STORE node in EntityVersion[v]
25         ElSE  /*elemento agregado*/
26              SET c to 1
27             FOR each chield /*para cada subatributo agregado*/
```

```
28                SET Name GET Name on the node
29                SET OriginalType GET datatype on the node
30                STORE Name in EntityVersion[v].chield[c]
31                STORE OriginalType in EntityVersion[v].chield[c]
32                IF OriginaType is "ATOMIC"
33                    STORE chield in EntityVersion[v].chield[c]
34                ElSE  /*elemento agregado*/
35                    INCREMENT c
36                    go to next chield
37                END IF
38            END FOR
39        END IF
40        IF EntityVersion[v] != EntityVersion[1..n] in EntitySchema /*se
              o esquema de entidade for diferente, adiciona mais uma versao
               de entidade*/
41            STORE EntityVersion[v] in Versions
42        ELSE
43            STORE EntityVersion[v] in EntitySchema
44
45        END IF
46        INCREMENT v
47        go to next node
48    END FOR
49    STORE Versions in SourceVersionSchema
50    STORE SourceVersionSchema in SourceSchema
51    END FUNCTION
52
53    FUNCTION generateSourceModel /*gera o esquema da fonte com todas as
          suas versoes*/
54    FOR each SourceVersionSchema in SourceSchema
55        GENERATE Source in model
56        READ SourceVersionSchema
57        FOR each Version in SourceVersionSchema
58            GENERATE SourceVersion in model
59            FOR each EntityVersion in Versions
60                GENERATE Entity in model
61                FOR each node in EntityVersion
62                    GENERATE Attribute in model
63                    IF node has child
64                        GENERATE SubAttributes in model
65                    ELSE
66                        go to next node
67                    END IF
68                END FOR
69            END FOR
70        END FOR
71    END FOR
```

```
72      END FUNCTION
73
74      FUNCTION generateEntityModel /*gera o esquema de entidade*/
75      FOR each EntityVersion in EntitySchema
76          GENERATE Entity in model
77          FOR each node in EntityVersion
78              GENERATE Attribute in model
79              IF node has child
80                  GENERATE SubAttributes in model
81              ELSE
82                  go to next node
83              END IF
84          END FOR
85      END FOR
86      END FUNCTION
87      /* o algoritmo gera dois arquivos diferentes dependendo da opcao do
           usuario*/
88      IF userOption = generateSourceModel
89          model <- generateSourceModel
90      ELSE
91          model <- generateEntityModel
92      END IF
93
94      RETURN model.gendb
95
96  END Identifier
```

Listing 4.1 – Algoritmo de engenharia reversa para identificação de esquemas versionados

A tarefa a ser executada pelo algoritmo consiste basicamente em transformar um arquivo XML (e suas respectivas tags/rótulos) em uma árvore também rotulada. A entrada é tratada como uma árvore rotulada que representa um arquétipo (modelo) de uma versão da fonte. O tratamento aplicado a cada objeto de entrada envolve uma análise de cada um de seus pares (n_i,v_i), mais especificamente, o tipo de dados do valor é verificado para identificar os elementos do modelo. Quando o valor de um par é um outro atributo ou um array de valores, esses objetos são recursivamente percorridos para identificar seus elementos internos (sub-atributos). O árvore alvo é formada por uma árvore de estrutura idêntica a da entrada, mas substituindo os valores dos nós intermediários por rótulos indicando o tipo de dado identificado. A Figura 13 ilustra essa transformação mostrando como a árvore (A) é convertida na árvore (B) que representa o esquema da versão fonte.

O algoritmo mantém uma coleção dos esquemas das instâncias presentes em cada arquivo e a cada nova instância analisada é verificado se a sua estrutura já existe nesta coleção. Se não existir, uma nova versão do esquema da Entidade é criada.

Após a engenharia reversa do esquema, uma transformação texto-para-modelo *T2M* é realizada para geração dos modelos sob o formato *.gendb* de acordo com o definido no

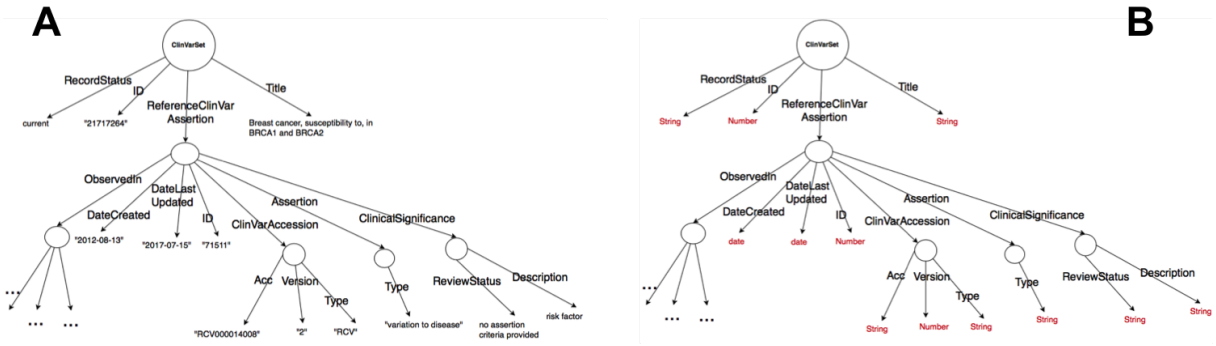

Figura 13 – Exemplo de transformação para identificação do esquema da instância de exemplo

algoritmo apresentado.

4.4.2 DEFINIÇÃO DO ALGORITMO DE GERAÇÃO DE ESQUEMAS PARA BASES DE DADOS ORIENTADAS A DOCUMENTO

A natureza organizacional das bases analisadas nesse trabalho apresentam uma aderência às chamadas bases de dados orientadas a agregados, pois uma base orientada a agregado armazena um conjunto de objetos semi-estruturados sem esquema fixo. Base de dados orientada a documento esta entre as bases classificadas como bases orientadas a agregados em por este motivo foi o tipo escolhido para a definição desta proposta. Assim, uma solução de MDE foi projetada para permitir a criação de bases de dados orientadas a Documento. Para esta solução foi definida uma transformação modelo-para-texto *M2T* que tem como entrada o modelo *.gendb* gerado e como saída o código *Javascript* de esquemas do MongoDB. O Algoritmo 4.2 apresenta o processo de transformação dos modelos GenDB para bases de dados Orientadas a Documento, especificamente esquemas do *MongoDB*. No MongoDB haverá uma coleção para cada versão de entidade (*Entity*), por exemplo.

```
1  Algorithm SchemaGenerator
2  Input: model.gendb
3  Output: genDBSchema.js
4
5  BEGIN
6      READ model.gendb
7      GENERATE genDBSchema    /*cria o esquema*/
8      FOR each Source in Model
9          Generate Collection in genDBSchema  /*cria colecoes*/
10         FOR each SourceVersion in Source
11             GENERATE Collection in genDBSchema
```

```
12              FOR each EntityVersion in SourceVersion
13                  GENERATE Collection in genDBSchema
14                  FOR each Attribute in EntityVersion
15                      IF Attribute has SubAttributes
16                          GENERATE Document in genDBSchema    /*cria
                               documentos*/
17                          FOR each SubAttributes in Document
18                              GENERATE NestedFields in genDBSchema    /*
                                   cria documentos aninhados*/
19                          END FOR
20                      ELSE
21                          GENERATE Attribute in genDBSchema   /*cria
                               atributos */
22                          go to next Attribute
23                      END IF
24                  END FOR
25              END FOR
26          END FOR
27      END FOR
28      RETURN genDBSchema.js
29  END SchemaGenerator
```

Listing 4.2 – Algoritmo para geração de esquemas de bases de dados orientadas a documento

O esquema define a estrutura dos dados armazenados em uma base do *MongoDB*. Esses esquemas são validados pelo MongoDB seguindo as especificações definidas no esquema da coleção. O validador do MongoDB verifica se os tipos de dados estão corretos e se todos os atributos obrigatórios foram informados, por exemplo.

4.5 CONSIDERAÇÕES

Este capítulo apresentou os esquemas identificados para os dados biológicos moleculares utilizados no processo de anotação de variantes genéticas. Em seguida apresentou o meta-modelo definido para representação formal destes esquemas bem como a definição da plataforma do domínio. Esta plataforma apresentou entre seus componentes, os algoritmos de transformação definidos para identificação de esquemas e geração de modelos, e para geração de esquemas de bases de dados orientadas a documento, além de uma linguagem de modelagem para visualização e edição de diagramas de esquemas.

É importante mencionar que o meta-modelo GenDB passou por rotinas de validações e verificações para chegar a versão apresentada nesse capítulo. Estas validações possibilitaram encontrar algumas falhas e realizar alguns refinamentos nas versões anteriores do meta-modelo GenDB. Como exemplo desses refinamentos pode-se mencionar a redução do número de metaclasses (de 12 para 7), pois a versão anterior apresentava um detalha-

mento de conceitos que não precisavam ser representados como metaclasses para atender aos requisitos de implementação do domínio. Outro exemplo de refinamento foi a adição de tipos de dados (*MaxCardinality* e *ValueType*) específicos para alguns atributos, o que permitiu uma representação mais adequada dos conceitos. Estes refinamentos permitiram que o meta-modelo mantivesse a representação dos conceitos do domínio com um número menor de elementos tornando-o mais simples e efetivo.

A linguagem de modelagem GenML foi definida seguindo os princípios de Moody (MOODY, 2010) para criação de notações visuais. Estes princípios permitiram que a linguagem atingisse uma boa anatomia para sua Sintaxe Visual (Concreta) composta de Símbolos Gráficos (Vocabulário Visual) adequados e Regras de Composição (Gramática Visual) claramente identificadas nos seus diagramas.

Na Tabela 7, exibe-se um comparativo das características presentes nos trabalhos analisados no Capítulo 3 com as características da abordagem proposta nesse trabalho. Esta comparação demonstra que combinação das técnicas e benefícios dessa proposta é única para o domínio sob investigação. Serão discutidas abaixo as principais características onde a abordagem proposta se difere dos demais trabalhos:

- Representação dos esquemas: A abordagem apresentada nesse trabalho propôs um meta-modelo para representação dos esquemas das bases de dados no contexto de MDE. A abordagem de MDE traz benefícios como, permitir a geração de modelos (diagramas) e esquemas de bancos de dados por meio da transformação de modelos (T2M e M2T). A meta-modelagem é útil para construir representações em alto nível de abstração permitindo que a informação seja uniformemente representada, o que favorece a qualidade do sistema em caraterísticas como interoperabilidade, extensibilidade, manutenibilidade ou reutilização. A existência de linguagens de meta-modelagem amplamente adotadas (por exemplo, Ecore) também fortalece o benefício do uso de meta-modelos.

- Modelos de dados: A abordagem apresentada nesse trabalho aproveitou as características de sistemas com esquemas flexíveis e de dados não uniformizados que não poderiam ser alcançados com bases de dados relacionais. Estas características são comuns a bases de dados semi-estruturados e bases NoSQl orientadas a agregados. Assim, esse trabalho apresentou soluções que possibilitam o gerenciamento de bases XML e dão suporte a criação de bases orientadas a documento.

- Evolução dos esquemas: A abordagem apresentada nesse trabalho propôs uma solução para o acompanhamento da evolução dos esquemas por meio da definição do que chamou-se de esquemas versionados. Este versionamento consiste em identificar e guardar todas as variações de estrutura apresentadas em entidades de mesmo tipo (que representam um mesmo conceito). Estas variações de estrutura também podem ser observadas nas fontes de dados (publicadores de dados de genoma) que

Abordagens	**Representação**	**Modelos**	**Evolução**	**Implementação**	**Modelagem**	**Geração**
(BALKO et al., 2004)	Esquema global	Relacional	–	Middleware	–	–
(LIU et al., 2009)	Esquema global	Relacional	–	Web Services	–	–
(LIU; LIU; YANG, 2010)	Esquema global	Relacional	–	Web Services	–	–
(MASSEROLI; CANAKOGLU; CERI, 2016)	Esquema global	Relacional	–	Data Wharehouse	–	–
GenDB	**Meta-Modelo**	**O. Documento**	**Versões**	**MDE**	**Linguagem de modelagem**	**transformações entre modelos**

Tabela 7 – Comparativo dos trabalhos acadêmicos relacionados com a abordagem proposta nesse trabalho.

lançam os conjuntos de dados periodicamente e, por este motivo, o versionamento dos esquemas das fontes de dados também foi abordado na proposta desse trabalho.

- Estratégia de Implementação: A abordagem apresentada nesse trabalho definiu suas soluções no contexto de MDE e irá fazer uso das ferramentas e recursos disponíveis nesse contexto para sua implementação. Entre esses recursos pode-se citar metamodelo, engenharia reversa, transformações de modelos e geração de códigos. O uso de técnicas de MDE permite agregar os benefícios já mencionados no tópico sobre representação dos esquemas. Mais detalhes sobre a implementação serão apresentados no Capítulo 5.

- Ferramentas de Modelagem: A abordagem proposta nesse trabalho definiu a linguagem de modelagem GenML com uma notação visual para edição e visualização dos esquemas em forma de diagramas. A implementação da ferramenta case será apresentada no Capítulo 5.

- Geração de códigos: A abordagem apresentada nesse trabalho propôs algoritmos para geração de modelos e esquemas de bancos de dados orientados a documento por meio da transformação entre modelos. A implementação desses algoritmos será apresentada no Capítulo 5.

No Capítulo 5 será apresentada a implementação de referência realizada para validação da abordagem proposta nesse capítulo.

5 IMPLEMENTAÇÃO DA ABORDAGEM PROPOSTA

> *"The theory of our modern technic shows that nothing is as practical as theory."*
>
> —Julius Robert Oppenheimer

Este capítulo apresenta a implementação de referência desenvolvida para validar a abordagem proposta nesse trabalho. Para esta implementação, foram desenvolvidas as soluções baseadas em modelos para engenharia reversa na identificação de esquemas de bases de dados (obtidos em forma de modelos), para criação de banco de dados a partir dos modelos dos esquemas obtidos. Também foi implementada uma ferramenta case para validação da linguagem de modelagem GenML apresentada no Capítulo 4.

A Tabela 8 apresenta um guia metodológico resumindo as etapas e atividades executadas para identificação das evidências e demais produtos que serão detalhados nesse capítulo.

Etapa de Execução do Processo - Fase: Implementação	
Objetivo	Concretizar em forma de artefatos de implementação as informações coletadas sobre o domínio e planejadas durante as etapas de Análise e Projeto
Atividades	(1) Implementação das soluções (2) Aplicação das soluções implementadas em ambiente real
Evidências	Ferramentas; Validação da abordagem proposta; Redução do tempo para a análise de dados de genoma

Tabela 8 – Guia metodológico para a etapa de Execução do Processo na fase de Implementação

5.1 IMPLEMENTAÇÃO DE REFERÊNCIA

Com a implementação de referência da plataforma do domínio foi possível a aplicação prática e a validação do meta-modelo, da linguagem de modelagem e dos algoritmos propostos nesse trabalho. Nesta plataforma o Meta-modelo forneceu um formalismo para representar os dados em um alto nível de abstração, e as transformações de modelos possibilitaram a geração de modelos e códigos de esquemas para bases de dados MongoDB.

A Figura 14 ilustra uma visão resumida do fluxo dos dados na implementação de referência desenvolvida nesse trabalho.

Primeiramente é realizada a identificação dos esquemas (*Squema Indentifier*) presentes nas fontes de dados (*Source*). Estes esquemas são transformados ($T2M$) para um

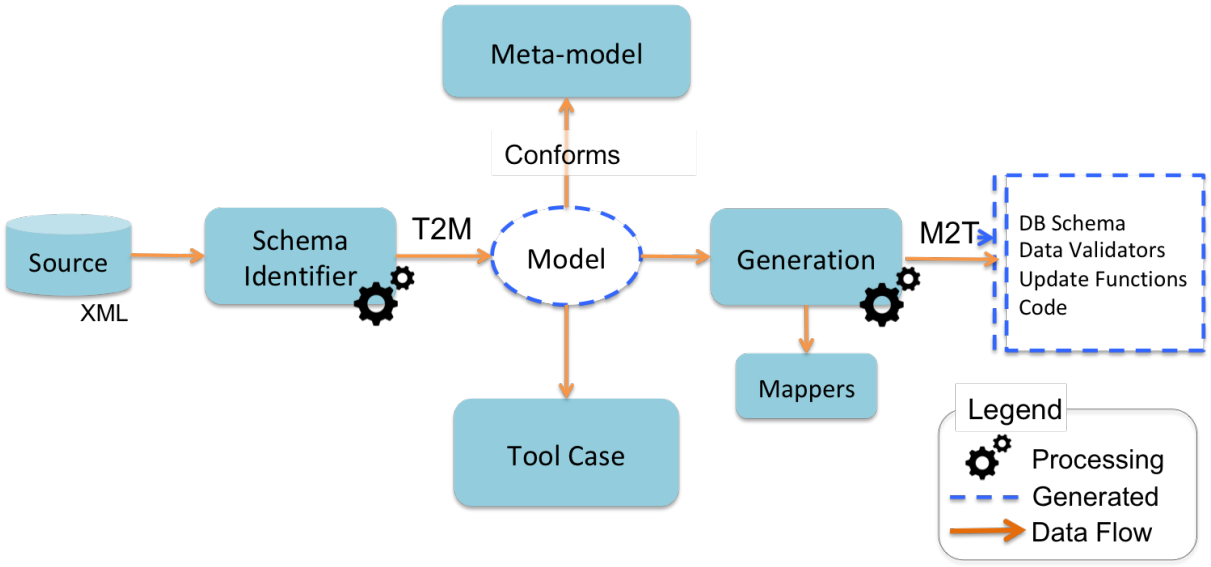

Figura 14 – Fluxo de dados na implementação de referência do domínio.

modelo(*Model*) que está em conformidade com nosso meta-modelo (*metamodel*) e pode ser visualizado na ferramenta (*Tool Case*), que será apresentada em detalhes na Seção 5.2. Tendo o modelo, é possível então realizar a segunda transformação (*M2T*) para geração do código de implementação da base de dados. Os códigos gerados para a criação das bases representam o esquema obtido pelo processo de identificação e estão de acordo com o meta-modelo.

As abordagens propostas nesse trabalho foram implementadas com o uso das ferramentas do *Eclipse Modeling Framework (EMF)* [1]. Este *framework* disponibiliza os recursos de modelagem e de geração de código para a construção de ferramentas e outras aplicações baseadas em meta-modelos.

A implementação da ferramenta Case será apresentada na Seção 5.2. A implementação dos algoritmos de identificação de esquema e criação das bases serão apresentados nas Seções 5.3 e 5.4, respectivamente.

5.2 LINGUAGEM DE MODELAGEM E FERRAMENTA CASE

Para a implementação da linguagem de modelagem GenML e da ferramenta *GenModelCASE* foi utilizada a ferramenta Sirius[2]. Sirius é uma ferramenta do *Eclipse Modeling Framework* para definir a sintaxe concreta (notação) de um meta-modelo existente.

A Figura 15 apresenta uma tela da ferramenta *GenModelCASE*. Esta é formada por cinco regiões distintas e cada uma destas regiões possui uma utilidade, como descrito a seguir.

[1] https://eclipse.org/modeling/emf/
[2] https://eclipse.org/sirius/

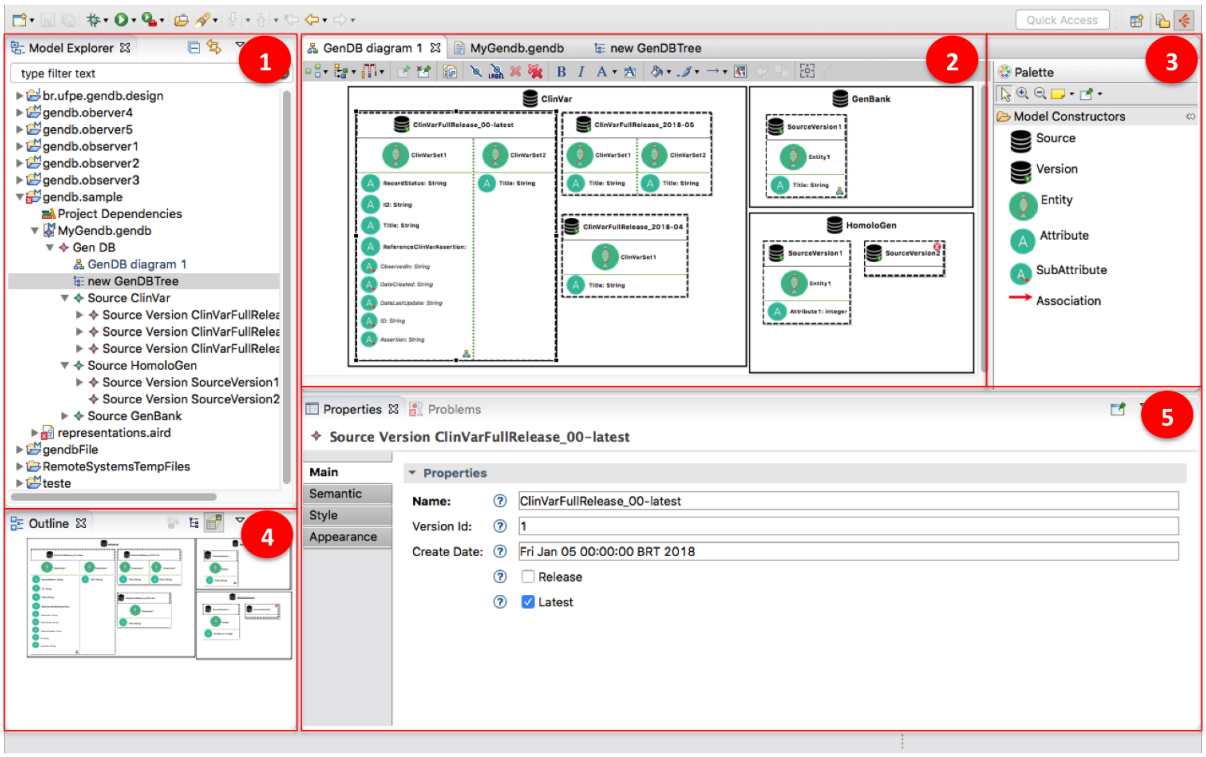

Figura 15 – *Workbench* da Ferramenta *GenModelCASE*

Na região, 1 exibem-se os projetos de modelagem. Na região 2, encontra-se a área de desenho, na qual é possível modelar a solução desejada. Na região 3, mostra-se a paleta de ferramentas, que contém as ferramentas necessárias para criação de cada construtor que podem ser utilizados na modelagem das soluções. Na região 4, encontra-se a área de *Outline* que facilita a navegação pelo modelo, principalmente quando este apresentar uma grande quantidade de objetos.

Na região 5, apresentam-se as propriedades (aba *Properties*) dos objetos selecionados na área de desenho.

Os avisos ou erros de modelagem detectados pela ferramenta podem ser vistos na aba *Problems* da região 5. Este erro é produto das validações, impostas pela sintaxe abstrata do meta-modelo GenDB, definidas para cada elemento do modelo e dispõe-se visualmente no formato de um pictograma característico de um erro em cor vermelha contendo um "X"em seu centro. Este pictograma pode ser visto no objeto (*SourceVersion2*) diagrama apresentado na Figura 15.

A Figura 16 apresenta um exemplo da aba *Problems*, indicando a necessidade do elemento Entity, uma vez que é obrigatória a presença de pelo menos uma entidade na versão da fonte (*SourceVersion*) para que o diagrama esteja de acordo com o meta-modelo GenDB.

A Figura 17 apresenta um exemplo da área de desenho com o diagrama em forma de árvore e as propriedades da entidade *ClinVarSet1*.

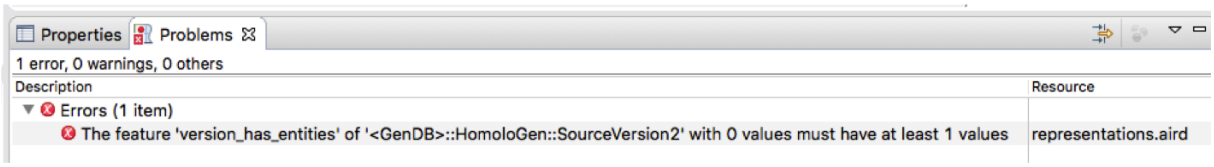

Figura 16 – Aba *Problems* da Ferramenta *GenModelCASE*

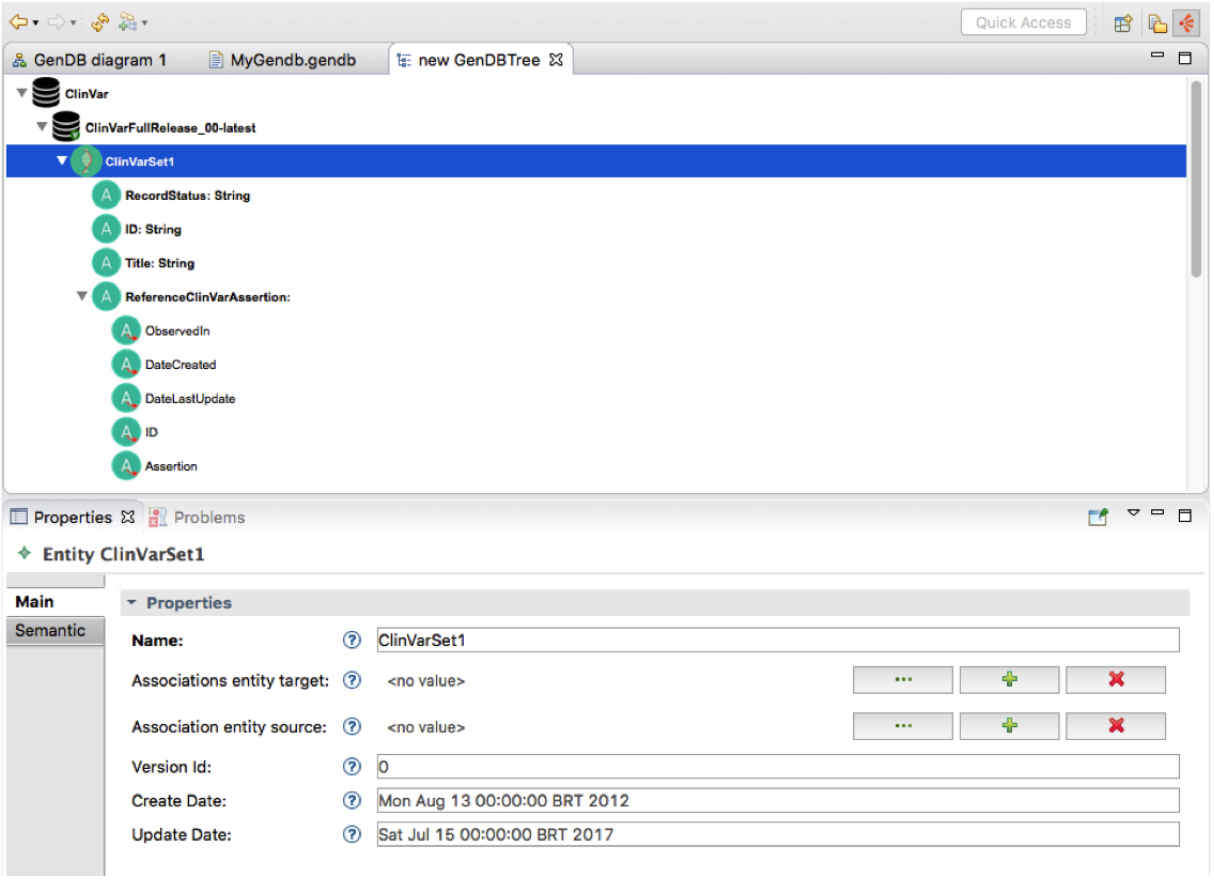

Figura 17 – Ferramenta *GenModelCASE* com diagrama em forma de árvore.

5.3 IDENTIFICAÇÃO DE ESQUEMAS

O algoritmo de identificação de esquemas foi implementado para identificar a estruturas das bases de dados e gerar modelos *.gendb*. Para que este modelo seja gerado de acordo com o esperado, o meta-modelo *GenDB* define a sintaxe abstrata (ou gramática) e a semântica (ou interpretação dos conceitos). Em resumo, este algoritmo é responsável por ler o arquivo da base em XML e gerar o modelo na extensão *.gendb*, interpretada pela ferramenta *GenModelCASE* para visualização dos diagramas do modelo gerado.

Nesta implementação, a linguagem *Xtext* foi utilizada para gerar uma DSL (gramática), com o objetivo de expressar o meta-modelo GenDB. Esta gramática é necessária para geração de modelos e código que estejam em conformidade com o meta-modelo GenDB.

A linguagem *Xtend* foi utilizada para implementar os geradores de modelo *.gendb*.

A Figura 18 ilustra a visão em XML do modelo gerado para uma entidade do exemplo.

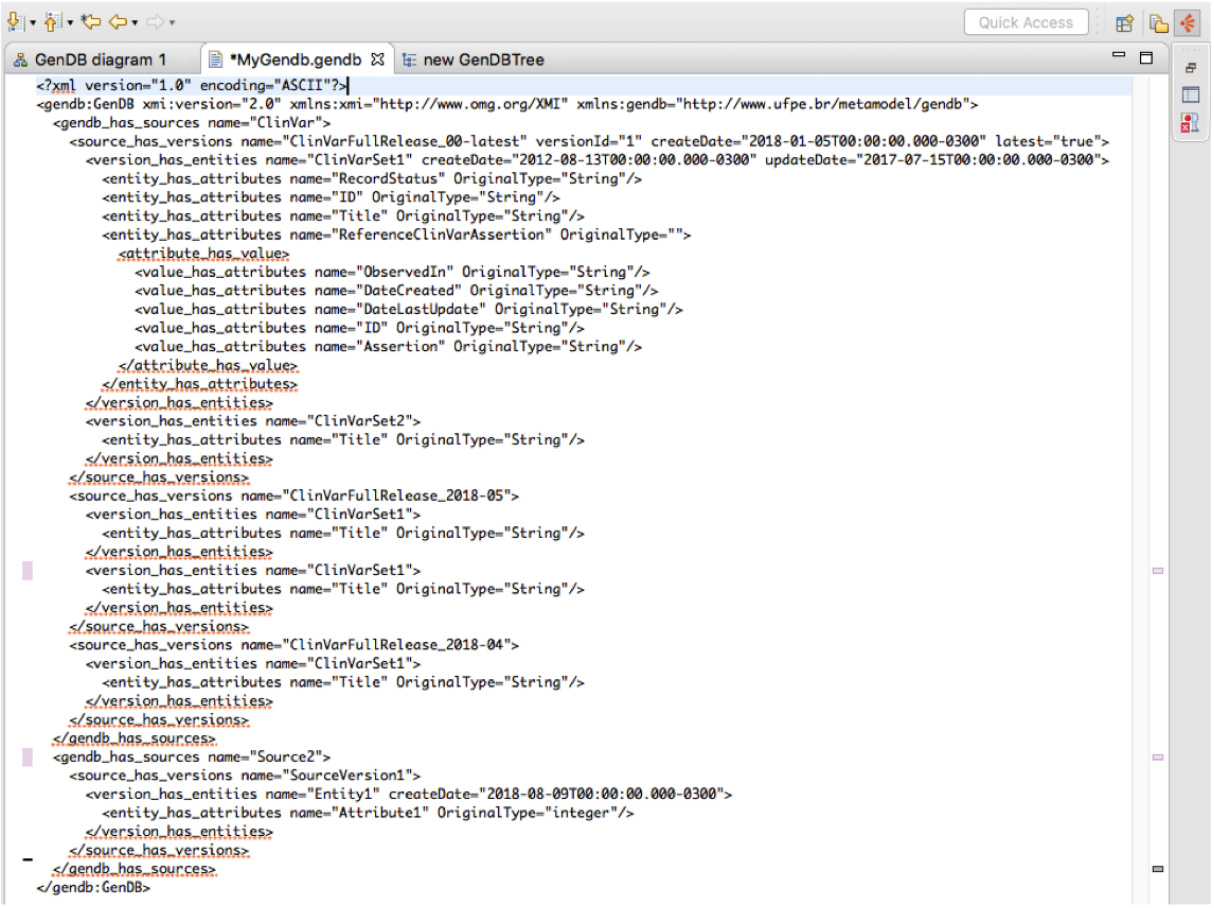

Figura 18 – Modelo .gendb gerado para o exemplo

5.4 CRIAÇÃO DE BASES DE DADOS

Este algotimo foi implementado com o uso de mapeadores *Object Document Mapper* (ODM) e inicialmente foi considerado o banco de dados NoSQL Orientado a documentos *MongoDB* para aplicação da solução proposta.

Neste algoritmo, uma transformação modelo-para-texto *M2T* foi implementada para gerar o esquema do modelo identificado. O ODM *Mongoose*[3] permite a geração de outros artefatos como validadores de dados, construção de consulta, entre outros. Como o artefato alvo desta pesquisa é principalmente o esquema do banco de dados, os demais artefatos não foram considerados para esta implementação.

Esta transformação pode ser implementada com o uso da linguagem *Xtend* mas nesta primeira versão do trabalho esta linguagem não foi utilizada nesta implementação.

[3] http://mongoosejs.com

A Figura 19 ilustra o esquema Mongoose para banco de dados MongoDB criado para a entidade do exemplo.

```
var ClinVarSet = new mongoose.Schema({
RecordStatus:{type: String},
ID: {type: Number},
Title: {type: String},
 ReferenceClinVarAssertion: {
     DateCreated: {type: Date},
     DateLastUpdated: {type: Date},
     ID: {type: Number},
     ClinvarAccession:{
         Acc: {type: String},
         Version: {type:Number},
         Type: {type: String},
    },
     Assertion: {
         Type: {type: String},
     },
     ClinicalSignificance: {
         ReviewStatus: {type: String},
         Description: {type: String},
    },
     ObservedIn: {
         Sample: {
            Origin: {type: String},
            Species: {type: String},
         },
     },
},
metadata:{
    name:{type:String, required: true},
    versionId{type:Number},
    createDate:{type:Date},
    udateDate:{type:Date},
}
});
```

Figura 19 – Esquema Mongoose para a entidade do exemplo

5.5 APLICAÇÃO CLIENTE E API

Além dos algoritmos e ferramentas apresentados anteriormente, também foi implementado um conjunto de soluções que permitisse ter uma prova de conceito da plataforma descrita no Capítulo 4. O principal objetivo desta implementação foi aplicar as técnicas definidas na plataforma em um ambiente real para verificar os possíveis impactos e justificar sua adoção. O ambiente utilizado para aplicação deste estudo foi um laboratório de sequenciamento de genoma humano. Para esta prova de conceito, foi necessário implementar uma aplicação cliente e uma API.

O software ClinGen foi a aplicação implementada para atuar como cliente da plataforma na realização de exames e diagnósticos. O sistema ClinGen foi desenvolvido pelo

laboratório parceiro e uma tela deste sistema está ilustrada na Figura 20. Esse sistema foi desenvolvido com *Django* e *Python*.

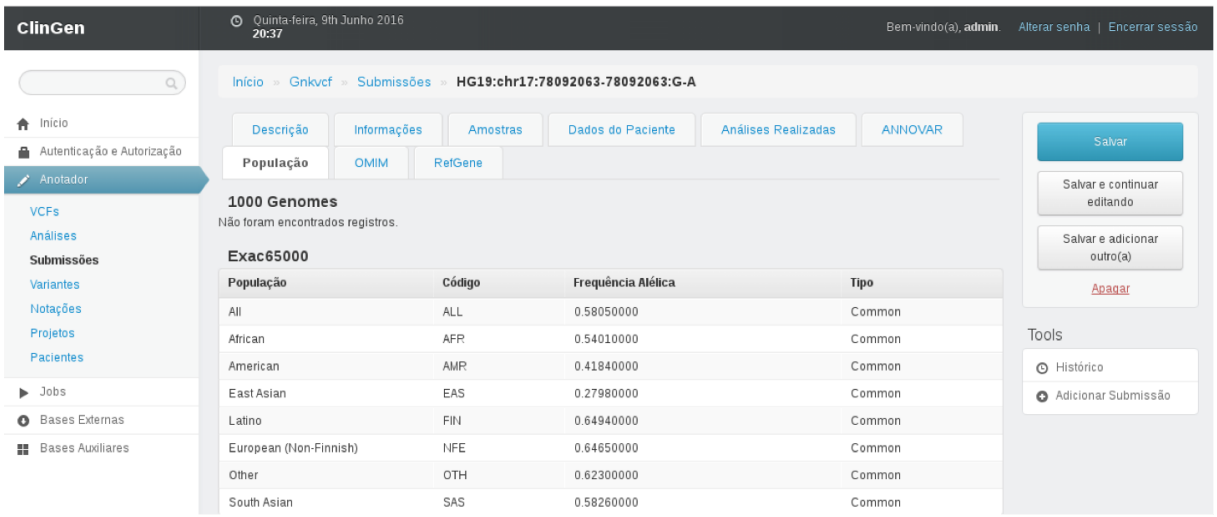

Figura 20 – Tela do sistema de anotação de variantes de genoma *ClinGen*.

O laboratório parceiro também desenvolveu uma API (*Application Programming Interface*) para acesso a base *GenDB* e recuperação dos dados para consumo dos aplicativos clientes. Uma imagem da API desenvolvida com *Django Rest Framework* esta ilustrada na Figura 21.

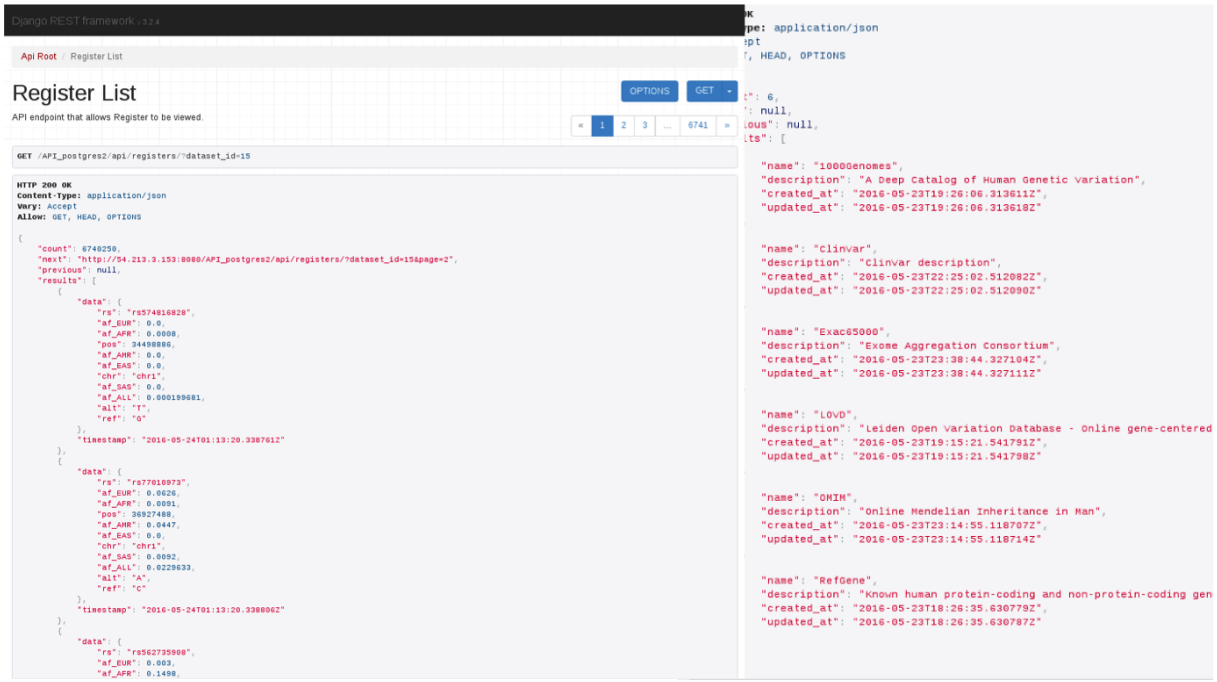

Figura 21 – API para acesso aos dados

Nesta primeira versão da implementação foi utilizado o tipo JSON do banco de dados *PostgreSQL* para armazenamento de amostras dos dados no formato de objetos JSON (*JavaScript Object Notation*) para a camada *Batch*.

5.6 CONSIDERAÇÕES

Com a implementação da abordagem proposta nesse trabalho foi possível validar os benefícios que a aplicação da plataforma provê para a análise de dados de genoma do laboratório. O principal impacto identificado na aplicação das soluções implementadas foi no tempo consumido para realização da análise dos dados. Para verificar este impacto um exame de exoma completo foi realizado. Antes da plataforma, o tempo de preparação dos dados pelo time da bioinformática era de aproximadamente 1 hora e a análise pelos especialistas tinha duração de 1 a 2 dias, resultando em um intervalo de 24 horas a 48 horas para diagnosticar apenas um paciente. Após adotar as estratégias propostas pela plataforma esses tempos reduziram para 30 minutos, 2 horas e 3 horas, respectivamente. É importante mencionar que o sistema ClinGen pode realizar uma nova análise quando uma atualização nas bases de dados ocorrer.

Os impactos observados no ambiente de implantação da solução foram suficientes para validação e justificativa da adoção das soluções pelo laboratório parceiro mas é importante reconhecer algumas limitações das soluções apresentadas nesse capítulo. Estas limitações serão abordadas nas pesquisas futuras que serão originadas desse trabalho. As principais limitações serão discutidas a seguir.

No que diz respeito a arquitetura, uma das suas limitações esta ligada a necessidade de melhorar a interoperabilidade entre as ferramentas implementadas pois ainda são necessárias muitas interações manuais para execução das soluções. Outra limitação é que a solução apresentada nesse trabalho não é capaz de realizar a migração dos dados. Esta atividade foi realizada de maneira semi-automatizada pela equipe do laboratório parceiro para garantir que o banco de dados histórico da camada *batch* fosse implementado e validado na arquitetura.

Quanto ao meta-modelo, este teve sua implementação e validação apenas para bases de dados XML (na identificação de esquemas) e orientadas a documento (na geração de esquemas), portanto se limita a publicadores e bancos de dados que trabalhem com estes formatos. Em relação a linguagem GenML, ha uma limitação quanto a escalabilidade dos diagramas, sendo necessárias estratégias para gerenciamento da complexidade dos diagramas considerando o número de esquemas de entidades que podem ser visualizados. Melhorias na usabilidade com a implementação de buscas e filtros podem diminuir esta limitação.

Com a implementação, também foi possível observar as vantagens de implementar estas soluções com a abordagem de MDE. Algumas das principais vantagens serão discutidas

a seguir:

Representação de informações em um alto nível de abstração. A meta-modelagem é um formalismo expressivo para representar as informações envolvidas em um processo de engenharia reversa de dados. Os modelos permitem que a informação seja uniformemente representada, o que favorece a qualidade do sistema em caraterísticas como interoperabilidade, extensibilidade ou reutilização. A existência de linguagens de meta-modelagem amplamente adotadas (por exemplo, Ecore) fortalece o benefício dos meta-modelos. Nesse trabalho definimos um meta-modelo para representar esquemas das bases de dados biológicos moleculares. Este meta-modelo também foi aplicado na criação de gramáticas e analisadores sintáticos necessários para geração de códigos e modelos.

Geração de Códigos. Um dos principais benefícios da MDE é permitir a geração de código para modelos, ferramentas de software e esquemas de bancos de dados. Nesse trabalho foram criadas cadeias de transformação de modelo para gerar os modelos a partir da ferramenta de inferência desenvolvida. Por sua vez estes modelos foram visualizados na ferramenta de modelagem gerada a partir da sintaxe concreta do meta-modelo e podem ainda ser utilizados para geração esquemas de bancos de dados.

Ferramental para construção de DSLs. Várias ferramentas de definição de DSL baseadas em meta-modelo estão disponíveis. Xtext, Xtend e Sirius são três das ferramentas mais amplamente utilizadas na plataforma Eclipse, Xtext e Xtend para DSLs textuais e Sirius para DSLs gráficas. Nesse trabalho o Xtext foi utilizado para criar uma gramática com o objetivo de expressar o meta-modelo. Esta gramatica foi necessária para geração de modelos e código. Por sua vez, Xtend foi utilizado para implementar os geradores de modelo gendb. Sirius foi usada para definir a DSML GenML com uma notação gráfica destinada a representar os diagramas de esquemas definidos para as bases de dados utilizadas na anotação de genoma.

Independência de plataforma. Independência de tecnologias de origem e destino pode ser alcançada com o uso de modelos unificados. Nesse trabalho o meta-modelo foi desenvolvido para representar esquemas independente de uma base XML específica. Levanta-se uma hipótese de que este meta-modelo poderia ser utilizado também para representação de esquemas de bases disponíveis em outros formatos mas é importante mencionar que esta hipótese não foi verificada no escopo deste trabalho. Da mesma maneira, não foi verificado se os modelos gerados a partir deste meta-modelo podem ser transformados para outras bases NoSQL ou Relacionais.

Algumas limitações também foram observadas na implementação das soluções com o uso da abordagem de MDE. Estas serão discutidas a seguir:

Rigidez. O uso de técnicas de MDE pode trazer muita rigidez ao projeto pois os modelos são flexíveis apenas quando esta flexibilidade é projetada, em pontos pré-definidos por expressões ou linguagens de baixo nível. Soluções baseadas em MDE exigem pouco esforço

de programação por trabalharem sempre em um alto nível de abstração. Em soluções de MDE a flexibilidade é possível apenas nas partes da solução cobertas pela DSL usada. Quanto maior o nível da DSL, mais pontos em comum são codificados no *framework* ou no mecanismo usado.

Pouca Personalização. Para projetar soluções com o uso de MDE é necessário entender as limitações das ferramentas pois estas apresentam algumas limitações para construção de soluções personalizadas. Para personalizar as soluções é exigido muito trabalho.

Versionamento. O versionamento de modelos gráficos ainda é um campo pouco pesquisado e os controladores de versão trabalham melhor com linguagens textuais. Assim, a maioria das ferramentas da abordagem de MDE não incluem cobertura para controladores de versão dificultando o trabalho colaborativo com times grandes.

No Capítulo 6 serão apresentados os métodos empíricos utilizados para avaliação das abordagens propostas nesse trabalho.

6 AVALIAÇÃO E RESULTADOS

> *"Science is facts; just as houses are made of stone, so is science made of facts; but a pile of stones is not a house, and a collection of facts is not necessarily science."*
>
> —Jules Henri Poincaré

Neste capítulo, serão apresentados os métodos empíricos utilizados para avaliação da abordagem proposta nessa tese.

Validar a adoção de toda a arquitetura da solução proposta neste trabalho é uma tarefa dificultosa pois causa mudanças em todo o fluxo de trabalho de uma companhia (laboratório). Esta atividade iria requerer que a arquitetura fosse implementada e executada por um longo período de tempo no mesmo laboratório, e isto é algo difícil de atender devido a várias restrições como limitações orçamentárias para investir em inovações, o que é uma situação comum em companhias brasileiras. Por esta rasão, pesquisadores e profissionais tendem a encontrar maneiras alternativas de avaliar e validar suas propostas. Com isso, técnicas como opinião de especialistas e estudos de caso podem ser muito úteis em prover *feedbacks* valiosos.

Diversas fontes de dados diferentes são normalmente usadas na pesquisa de estudos de caso. Os dados, qualitativos e quantitativos, oriundos de questionários e observação, desempenham um papel importante, pois oferecem informações valiosas sobre o caso (EASTERBROOK et al., 2007). Nesse sentido, a Seção 6.1 apresentará um questionário baseado em opinião de especialistas para avaliação do meta-modelo GenDB por meio da verificação de critérios e indicadores de qualidade. A Seção 6.2 apresentará uma observação participante para avaliação da linguagem de modelagem GenML.

A Tabela 9 apresenta um guia metodológico resumindo as etapas e atividades executadas para identificação das evidências que serão apresentadas neste capítulo.

Etapa de Análise de Resultados - Fase: Avaliação	
Objetivo	Analisar empiricamente os resultados obtidos nas etapas anteriores
Atividades	(1) Realizar um Survey para verificação da qualidade dos meta-modelo GenDB. (2) Realizar uma observação participante para avaliar a Ferramenta e Linguagem de Modelagem GenML.
Evidências	(1) Indicadores de qualidade do meta-modelo GenDB. (2) Medições sobre uso e compreensão da linguagem de modelagem GenML.

Tabela 9 – Guia metodológico para a etapa Análise de Resultados na fase de Avaliação

6.1 QUESTIONÁRIO BASEADO EM OPINIÃO DE ESPECIALISTAS

A abordagem para este estudo foi sistematicamente organizada em quatro fases. Na primeira fase, o objetivo era definir, avaliar e validar a pesquisa. Na segunda fase, os especialistas foram selecionados e contatados de acordo com as diretrizes que serão discutidas a seguir. Na terceira fase, a pesquisa foi enviada aos especialistas e os dados foram coletados. Finalmente, na quarta fase, os dados foram analisados com o objetivo de avaliar o Meta-modelo GenDB quanto à sua qualidade, com base na opinião de especialistas. As próximas seções discutem as definições do método utilizado, do planejamento e execução do estudo, e da análise dos resultados.

6.1.1 Opinião de Especialista

Opinião de Especialista pode ser definida como uma série de empreendimentos científicos que são empregados para interpretar dados, prever o comportamento de um sistema e avaliar incertezas (COOKE, 1991). Isto se trata de "especulações, suposições e estimativas de pessoas que são consideradas especialistas na medida em que estas servem como "*input* cognitivo" em algum processo de decisão" (LI; SMIDTS, 2003; COOKE, 1991).

O uso generalizado da opinião de especialistas decorre do fato de que o conhecimento em muitos dos campos envolvidos na análise probabilística e nos processos de tomada de decisão é geralmente raro e incompleto (LI; SMIDTS, 2003). Da mesma forma, a informação experimental ou estatística com base na qual as previsões ou decisões podem ser feitas não é facilmente disponível (LI; SMIDTS, 2003). A pesquisa atual em engenharia de software está fazendo uso constante da opinião de especialistas. No entanto, ainda há alguma controvérsia (KITCHENHAM, 2007) e ceticismo na comunidade científica em relação a este assunto, e nenhuma estrutura universalmente aceita para lidar com a opinião de especialistas está disponível (LI; SMIDTS, 2003). Parte da hesitação em aceitar o uso de opinião de especialistas decorre do alto grau de subjetividade em exercícios anteriores, em que a opinião de especialistas foi usada (LI; SMIDTS, 2003). Além disso, outros pesquisadores enfatizam o problema de confiar em evidências informais que podem ser influenciadas pela opinião pessoal (KITCHENHAM, 2007). Finalmente, de acordo com Li e Smidts (LI; SMIDTS, 2003), o número de especialistas necessários em um estudo, a identificação e calibração de viés e a técnica de agregação de opinião especializada são questões que precisam ser esclarecidas antes que um estudo que usa a opinião de especialistas possa ter sucesso. Nas próximas seções, essas questões serão discutidas em detalhes.

6.1.1.1 O Número de Especialistas

A priori, se um especialista é perfeito (ou seja, ele / ela tem conhecimento infinito sobre o tópico em estudo e nunca erra), o número de especialistas necessários para um processo de elicitação de especialistas é um. No entanto, há uma tendência de buscar tantos especialistas quanto possível, justificado por uma percepção de segurança nos números. Quando consideramos a universidade de especialistas, podemos notar que existem muitas fontes de dependência entre especialistas, ou seja, semelhanças em treinamento, educação e experiência. A pesquisa mostrou que, se os especialistas são dependentes, a redução da incerteza[1] se aproxima de um valor limitante, tornando desnecessário o uso de muitos especialistas (LI; SMIDTS, 2003). Portanto, esse resultado pode ser de valor monumental na prática, porque significa que os exercícios de julgamento por especialistas podem ter sucesso com um número menor de especialistas.

Li e Smidts (LI; SMIDTS, 2003) discutiram a diferença entre o papel do especialista em um processo de opinião especializada e o papel de uma amostra usada em um cálculo estatístico. Para eles, a elicitação de opinião de especialistas é "a aquisição do conhecimento de especialistas do mundo real e, se o especialista for credível, até mesmo um especialista é adequado para a obtenção do valor real". Além disso, se os especialistas forem dependentes, o aumento do número de especialistas não ajudará a melhorar a precisão ou a credibilidade dos assuntos elicitados. Por outro lado, os dados de amostragem representam uma instância probabilística de um fenômeno e a precisão da descrição melhora, aumentando o tamanho da amostra. Para este trabalho, de acordo com a questão das dependências de especialistas, foram selecionados 4 especialistas, compondo o grupo de especialistas elicitados na pesquisa. Eles representam papéis confiáveis na comunidade de bioinformática, de acordo com alguns critérios predefinidos. A próxima seção descreve o processo de seleção e os critérios.

6.1.1.2 Seleção dos Especialistas neste Estudo

Em um contexto ideal, os especialistas devem ser escolhidos de acordo com estimativas ou julgamentos mais precisos (NUREG-1150, 1989). No entanto, não existe um padrão bem definido e conhecido (LI; SMIDTS, 2003) de como exatamente isso pode ser alcançado, para que possamos formular um critério que possa ser usado para sistematizar o processo de seleção. NUREG (NUREG-1150, 1989) apresenta um conjunto de diretrizes para a seleção de especialistas:

- Os especialistas devem ter demonstrado experiência em publicações, experiência prática e consultoria ou gestão de pesquisas nas áreas relacionadas às questões em estudo;

[1] Incerteza aqui quer dizer a diferença entre a entrada do especialista e o valor verdadeiro. Essa diferença é uma variável estocástica.

Especialistas	**Ocupação**
Especialista 1	Academia e Indústria
Especialista 2	Academia e Indústria
Especialista 3	Indústria
Especialista 4	Indústria

Tabela 10 – Lista dos especialistas que participaram do estudo

- Cada especialista deve ser versátil o suficiente para ser capaz de abordar vários problemas e ter uma vasta experiência para considerar como esses problemas seriam usados;
- Os especialistas devem representar uma ampla variedade de experiências obtidas em universidades, empresas, laboratórios ou agências governamentais;
- Os especialistas devem representar uma perspectiva tão ampla quanto possível do assunto; e,
- Os especialistas devem estar dispostos a ser induzidos sob a metodologia a ser usada.

Foram selecionados 4 especialistas com base nas diretrizes anteriores. Os principais requisitos para o estudo foram conhecimento e experiência no processo de análise de dados de genoma. Além disso, de acordo com as recomendações do NUREG (NUREG-1150, 1989), especialistas da indústria e do meio acadêmico com diferentes níveis de experiência foram selecionados. A Tabela 10 apresenta uma lista resumida dos especialistas selecionados.

6.1.1.3 Viés dos Especialistas e Calibração neste Estudo

Tversky e Kahneman (TVERSKY; KAHNEMAN, 1974) dividiram vieses de especialistas em duas classes: viés de localização e viés de excesso de confiança. O primeiro refere-se à superestimação sistemática da quantidade variável, e o segundo refere-se a uma tendência de indicar um intervalo artificialmente menor do que o estado real de conhecimento dos especialistas. A avaliação e compensação desses vieses pelo analista são conhecidas como calibração de especialistas. Um método adicional para reduzir os vieses de excesso de confiança é encorajar os especialistas a encontrar razões que contradigam suas opiniões iniciais. Os vieses de localização são corrigidos por um método de agregação bayesiano (LI; SMIDTS, 2003).

Neste trabalho, foi definido o processo de levantamento de opinião de especialistas de forma que os estes pudessem fornecer explicações detalhadas para suas classificações. Além disso, eles também foram contatados para esclarecer pontos de dúvida e nenhum viés óbvio foi encontrado.

6.1.1.4 Agregação da Opinião dos Especialistas neste Estudo

Li e Smidts (LI; SMIDTS, 2003) afirmaram que, quando métodos de agregação são usados, eles variam de métodos fáceis de usar, como a simples média de várias entradas de especialistas para técnicas mais complexas, como o modelo clássico de métricas (CHIDAMBER; KEMERER, 1994), a Agregação Bayesiana (CHHIBBER; APOSTOLAKIS; OKRENT, 1992) e a Análise Fatorial (KIM; MUELLER, 1978a; KIM; MUELLER, 1978b). Alguns estudos em outras áreas, como a previsão de tempo, demonstram que a agregação de opiniões de especialistas é necessária, uma vez que indicam que as opiniões agregadas, mesmo que obtidas apenas por uma média simples, são consistentemente melhores que as opiniões de especialistas individuais (SCHNAARS, 1986). Médias aritméticas e geométricas, por exemplo, são dois métodos de agregação usados na prática. Esses métodos pressupõem que todos os especialistas sejam igualmente ponderados, o que é uma forte suposição. No entanto, isso não os impediu de serem amplamente utilizados na prática (LI; SMIDTS, 2003).

Neste trabalho, será adorado o método de agregação por análise fatorial, uma técnica de redução de dados utilizada para reduzir uma grande quantidade de variáveis observadas a um número menor de fatores. Além disso, todos os especialistas foram igualmente ponderados durante a agregação, uma vez que não foi observada nenhuma diferença significativa em termos de credibilidade e importância dos especialistas. A execução e análise dos resultados da análise fatorial serão apresentados nas Seções 6.1.2.3 e 6.1.3, respectivamente.

6.1.2 Planejamento e Execução do Estudo

6.1.2.1 Contexto e Variáveis Medidas

O questionário foi aplicado para avaliar o meta-modelo e gerar indicadores de qualidade por meio da verificação de critérios de qualidade. Os critérios analisados para cada indicador de qualidade são: Qualidade Empírica (Compreensibilidade e Simplicidade); Qualidade Sintática (Corretude e Integridade); Qualidade Semântica (Completude, Validade, Integração, Reuso e Minimalismo); Qualidade Pragmática (Compreensibilidade); e Qualidade Organizacional (Flexibilidade, Integração e Implementabilidade).

6.1.2.2 Coleta de Dados: o Questionário

A coleta de dados é sempre realizada em relação a uma unidade de análise bem definida. Como mencionado anteriormente, nesta avaliação a unidade de análise será composta da *persona*: Especialista. Para este estudo tivemos a colaboração de quatro especialistas do laboratório parceiro Genomika [2].

[2] https://www.genomika.com.br

Este trabalho fez uso da técnica de coleta de dados: questionário. Esta técnica foi escolhidas por permitir a coleta de dados tanto qualitativos quanto quantitativos além de se apresentarem como instrumentos tradicionais para o tipo de coleta de dados que se pretende realizar. O questionário é a técnica de coleta de dados comumente utilizado em entrevistas e experimentos que visam coletar opiniões de usuários e especialistas (EASTERBROOK et al., 2007).

O questionário completo utilizado para coleta dos dados nesta avaliação está disponível no Apêndice A.

6.1.2.3 Técnica de Análise dos Dados

Para se chegar a um indicador de qualidade foi utilizado o método estatístico de análise fatorial (KIM; MUELLER, 1978a; KIM; MUELLER, 1978b). A análise fatorial é uma técnica de redução de dados utilizada para reduzir uma grande quantidade de variáveis observadas a um número menor de fatores. Essa redução se baseia no padrão de correlação observado entre as variáveis, e é explicada (representada) pelos fatores (KIM; MUELLER, 1978a; KIM; MUELLER, 1978b). Neste trabalho os fatores são os indicadores de qualidade e as variáveis observadas são os critérios de cada indicador.

Inicialmente foi necessário identificar o padrão de correlação (força) entre as variáveis observadas. A literatura recomenda que a maior parte dos coeficientes seja superior a 0,3, independente do sinal para indicar adequabilidade dos dados para a técnica de redução (KIM; MUELLER, 1978b). Como a análise fatorial depende do padrão de correlação entre as variáveis observadas, espera-se que variáveis estatisticamente independentes não contribuam para a construção de um fator, logo, o pesquisador pode optar por excluir uma variável e estimar novamente a análise fatorial. A técnica escolhida para a análise fatorial foi a análise de componentes principais (KAPLUNOVSKY, 2009), visto que queremos gerar um resumo empírico do conjunto de dados. Após verificar a adequabilidade dos dados foi realizada a extração de fatores para identificar quantos fatores podem ser gerados a partir dos dados. A técnica utilizada para a extração foi a regra do *eigenvalue* (critério de *Kaiser*) (GARSON, 2018) que sugere que devem ser extraídos apenas os fatores com valor de *eigenvalue* acima de um (1). O tipo de rotação de fatores escolhido foi a rotação ortogonal *Varimax* (PALLANT, 2013) por ser o mais comumente utilizado. Este método procura minimizar o número de variáveis que apresentam altas cargas fatoriais em cada fator. O principal objetivo da rotação dos fatores é tornar o resultado empírico encontrado mais facilmente interpretável, conservando as suas propriedades estatísticas. Assim, a rotação tem o objetivo de facilitar a visualização da relação entre as variáveis observadas e os componentes extraídos. Quando apenas um fator é extraído a solução não pode ser rotacionada.

Em resumo, o planejamento da análise fatorial é realizado em três estágios:

- Verificar a adequabilidade da base de dados;
- Determinar a técnica de extração e o número dos fatores(componentes) a serem extraídos; e
- Decidir o tipo de rotação dos fatores (componentes).

A base para a criação do índice está na matriz de componentes. Utilizando-se dos valores desta matriz, cria-se primeiramente o índice de qualidade de cada fator para cada especialista, onde cada variável que compõe esse fator é ponderada pela sua carga fatorial. A resposta de cada especialista para uma dada variável é considerada como um caso (ocorrência) da variável na análise fatorial. Assim, o índice de qualidade no fator K de um especialista qualquer é definido de acordo com a equação:

$$IQ\left(F_{kn}\right) = \frac{\sum_{i=1}^{j}(|p_i|.x_i)}{\sum_{i=1}^{j} p_i} x100\%$$

onde:

- $IQ\left(F_{kn}\right)$ é o índice de qualidade do especialista n no fator k;
- j é o número de variáveis no fator k;
- $|p_i|$ é o módulo da carga fatorial da variável i no fator k; e
- x_i é a nota da variável i no fator k.

Como mencionado anteriormente, após a execução da análise o índice de cada fator (escore) é salvo como uma variável ($IQ\left(F_{kn}\right)$) que apresenta um valor para cada caso observado. Para se chegar ao indicador de cada critério de qualidade é necessário normalizar e reduzir os valores de todos os escores a um único valor. Para tal, os dados precisam ser normalizados para uma escala padronizada, por exemplo um intervalo de 0 a 1. A normalização é realizada pela fração da variação total de acordo com a equação:

$$Z = \frac{IQ(F_{kn}) - \min IQ(F_k)}{\max IQ(F_k) - \min IQ(F_k)}$$

onde:

- $IQ\left(F_{kn}\right)$ é o índice de qualidade do especialista n no fator k (valor que se pretende normalizar);
- $\min IQ\left(F_k\right)$ é o menor valor do índice de qualidade no fator k;
- $\max IQ\left(F_k\right)$ é o maior valor do índice de qualidade no fator k.

Após a normalização, a média aritmética dos escores é realizada para se chegar ao valor do indicador. É importante mencionar que os indicadores não são comparáveis entre si por possuírem configurações diversas.

A seguir serão apresentados os dados extraídos na coleta e as implementação e resultados da análise fatorial obtidas para cada critério de qualidade.

6.1.2.4 Dados Coletados

A Figura 22 ilustra as notas dos critérios para cada especialista. Estas notas foram obtidas pela indicação dos especialistas em uma escala *Likert* com pontuação variando de 1 a 5. Em todos os casos os critérios tiveram pontuação entre 4 e 5 indicando que o meta-modelo possui um bom nível de satisfação dos especialistas.

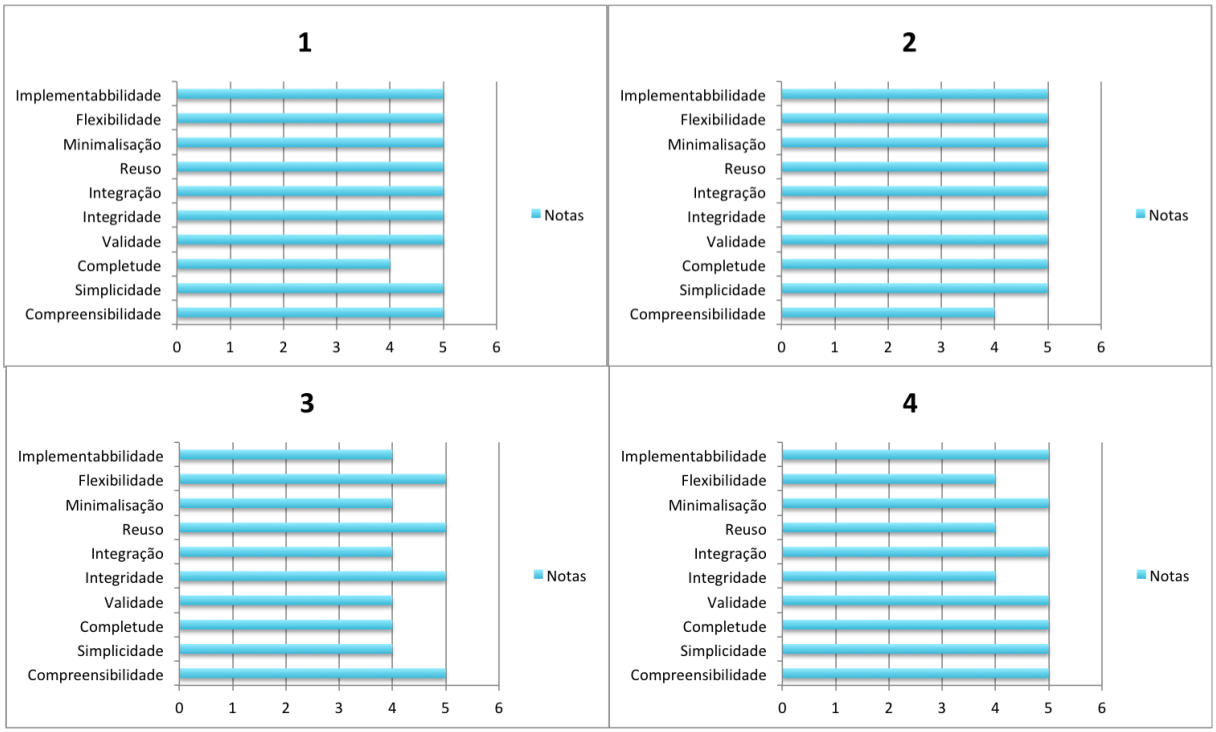

Figura 22 – Dados brutos dos critérios/variáveis para cada especialista (1 a 4)

6.1.3 Análises e Interpretação dos Resultados Baseada no Método GQM

Para se definir um estudo, este deve ser conduzido de acordo com um objetivo específico e um conjunto de questões que representem a definição operacional do objetivo. Assim, o método GQM (*Goal, Question, Metrics*, em português Objetivo/Questão/Métricas) (BASILI; CALDIERA; ROMBACH, 1994) foi usado para definir o objetivo desta análise e

para derivar as questões que devem ser respondidas para determinar se o objetivo foi alcançado ou não.

Por várias razões, um objetivo é definido para um objeto com relação a atributos de qualidade, de vários pontos de vista e relativos a um ambiente específico. Exemplos de objeto de medições são projetos, métodos e tecnologias (WOHLIN et al., 2000). Assim, o objetivo desta análise foi definido da seguinte forma:

Objetivo: Analisar o meta-modelo GenDB quanto à avaliação de qualidade do ponto de vista dos especialistas no contexto do laboratório parceiro.

Depois de definir o objetivo, ele é então refinado em questões que tentam caracterizar os objetos de medição com relação a um indicador de qualidade selecionado e determinar sua qualidade a partir do ponto de vista selecionado. Neste trabalho teremos uma questão para cada indicador de qualidade:

Q1: *Qual o nível de Qualidade Empírica do meta-modelo GenDB de acordo com a satisfação dos especialistas em relação aos critérios Compreensibilidade e Simplicidade?*

Q2: *Qual o nível de Qualidade Semântica do meta-modelo GenDB de acordo com a satisfação dos especialistas em relação aos critérios Completude, Validade, Integridade, Reuso e Minimalização?*

Q3: *Qual o nível de Qualidade Organizacional do meta-modelo GenDB de acordo com a satisfação dos especialistas em relação aos critérios Flexibilidade, Integração e Implementabilidade?*

Q4: *Qual o nível de Qualidade Pragmática Social do meta-modelo GenDB de acordo com a satisfação dos especialistas em relação ao critério Compreensibilidade?*

Em seguida, um conjunto de métricas é associado a cada pergunta para respondê-la de forma quantitativa, objetiva ou subjetivamente, conforme definido por Wohlin et al. Neste trabalho, as métricas (variáveis observadas) serão os critérios considerados para cada indicador de qualidade. Para o indicador de Qualidade Empírica serão observados os níveis indicados para os critérios Compreensibilidade e Simplicidade, por exemplo. Como mencionado anteriormente, estes critérios foram medidos e agregados usando o método estatístico de analise fatorial para geração dos indicadores de qualidade.

Visto isso, as análises e interpretação dos resultados das avaliações serão apresentadas para cada indicador de qualidade a seguir.

6.1.3.1 Qualidade Empírica - Q1

O indicador de Qualidade Empírica busca sinalizar como o meta-modelo se apresenta em termos de ergonomia cognitiva, ou seja, como seus elementos e estrutura são compreendidos em relação ao domínio o qual este representa. Diante disso, foram observados os níveis indicados pelos especialistas para as variáveis Compreensibilidade e Simplicidade. A Figura 22 apresentou as notas indicadas pelos especialistas para estas variáveis. Em ambas as variáveis a média das notas foi de $4,75$, indicando um ótimo nível de satisfação

para estes critérios do meta-modelo, uma vez que a nota máxima esperada é 5. Para reforçar a validade deste indicador estas notas foram analisadas estatisticamente seguindo o método de análise fatorial descrito anteriormente.

Na verificação da adequabilidade dos dados, a correlação entre as variáveis foi de $0,333$, ficando muito próxima do mínimo de 0,3 indicado na literatura (Figura 23 - Matriz de correlações). O outro teste realizado para a inclusão ou exclusão de variáveis é o nível de associação entre a variável e o fator extraído, sinalizado pelo valor da comunalidade. A comunalidade consiste na correlação com as outras variáveis e também com o indicador que está sendo criado. Todas as variáveis apresentaram valor de comunalidade $0,667$, ficando acima do mínimo de 0,4 indicado na literatura (Figura 23 - Comunalidades). Em ambos os casos, os testes sugeriram que os dados são adequados à análise fatorial, mesmo que com um baixo valor de correlação entre as variáveis.

O passo seguinte é determinar o número de fatores (componentes) que serão extraídos. Na Figura 23 (Variância Total Explicada) o critério de *Kaiser* sugere que deve-se extrair apenas um fator (componente) que apresenta um *eigenvalue* de 1,333, carregando 66,667 % da variância das variáveis originais.

Matriz de correlações

		Compreensibilidade	Simplicidade
Correlação	Compreensibilidade	1.000	-.333
	Simplicidade	-.333	1.000

Comunalidades

	Inicial	Extração
Compreensibilidade	1.000	.667
Simplicidade	1.000	.667

Método de Extração: análise de Componente Principal.

Variância total explicada

	Autovalores iniciais			Somas de extração de carregamentos ao quadrado		
Componente	Total	% de variância	% cumulativa	Total	% de variância	% cumulativa
1	1.333	66.667	66.667	1.333	66.667	66.667
2	.667	33.333	100.000			

Método de Extração: análise de Componente Principal.

Figura 23 – Análise fatorial para o componente Qualidade Empírica

Como mencionado anteriormente, o índice de cada fator (escore) é salvo como uma variável que apresenta um valor para cada caso observado. Para se chegar ao indicador do critério, os escores foram normalizados para uma escala padronizada que vai de 0 a 1. Após a normalização, a média aritmética dos escores foi realizada para se chegar ao valor do indicador. A Figura 24 ilustra a estatística descritiva com o resultado final da análise para o indicador de Qualidade Empírica que possui uma média de 50%. Este resultado indica um nível satisfatório de Qualidade Empírica, considerando que as variáveis apresentaram um baixo valor de correlação e que o fator extraído carrega apenas 66,667% da

variância das variáveis.

Estatística Descritiva

	N	Mínimo	Máximo	Média	Desvio
Indice_Q_Empirica	4	.00	1.00	**.5000**	.40983
N válido (de lista)	4				

Figura 24 – Índice de Qualidade Empírica

6.1.3.2 Qualidade Semântica - Q2

O indicador de Qualidade Semântica busca sinalizar como o meta-modelo reflete o conhecimento relevante para o domínio ao qual este representa. Diante disso, foram observados os níveis indicados pelos especialistas para as variáveis Completude, Validade, Integridade, Integração, Reuso e Minimalização. Figura 22 apresentou as notas indicadas pelos especialistas para todas as variáveis. Para a variável Completude, a média das notas foi de $4,5$. Para todas as demais variáveis a média foi de $4,75$. Este resultado indica um ótimo nível de satisfação para estes critérios do meta-modelo, uma vez que a nota máxima esperada é 5. Para reforçar a validade deste indicador, estas notas foram analisadas estatisticamente seguindo o método de análise fatorial.

Na verificação da adequabilidade dos dados, as correlações entre as variáveis superaram o mínimo de 0,3 (na Figura 25 - Matriz de correlações, pode-se observar os valores $0,577$ e $0,333$ de correlação). Logo os dados são adequados a utilização da análise fatorial e nenhuma variável é estatisticamente independente das demais. O teste final para a inclusão ou exclusão de variáveis é o nível de associação entre a variável e o fator extraído, sinalizado pelo valor da comunalidade. A comunalidade consiste na correlação com as outras variáveis e também com o indicador que está sendo criado. Todas as variáveis ficaram acima do mínimo de 0,4 (Figura 25 - Comunalidades). Em ambos os casos, os testes sugerem que os dados são adequados à análise fatorial.

O passo seguinte é determinar o número de fatores(componentes) que serão extraídos. O critério de Kaiser sugeriu que devem ser extraídos dois fatores: o primeiro apresenta um *eigenvalue* de 4.000, carregando 66,667% da variância das variáveis originais. O segundo fator apresenta *eigenvalue* de 1,577, carregando 26,289% da variância. Em conjunto, esses dois fatores explicam 92,956% da variância das variáveis originais. Estes dados estão ilustrados na Figura 25 - Variância Total Explicada.

A decisão tomada aqui foi realizar a análise considerando dois componentes para o indicador de Qualidade Semântica (QS): (1) um com as variáveis Completude, Validade, Integração e Minimalisação, para um componente de QS referente ao conhecimento específico do domínio e (2) outro com as variáveis Reuso e Integridade, para um componente de QS referente a aplicabilidade do meta-modelo. A integridade é interessante para este segundo

Matriz de correlações[a]

		Completude	Validade	Integridade	Integração	Reuso	Minimalisação
Correlação	Completude	1.000	.577	-.577	.577	-.577	.577
	Validade	.577	1.000	-.333	1.000	-.333	1.000
	Integridade	-.577	-.333	1.000	-.333	1.000	-.333
	Integração	.577	1.000	-.333	1.000	-.333	1.000
	Reuso	-.577	-.333	1.000	-.333	1.000	-.333
	Minimalisação	.577	1.000	-.333	1.000	-.333	1.000

a. Essa matriz não é positiva definida.

Variância total explicada

	Autovalores iniciais			Somas de extração de carregamentos ao quadrado			Somas de rotação de carregamentos ao quadrado		
Componente	Total	% de variância	% cumulativa	Total	% de variância	% cumulativa	Total	% de variância	% cumulativa
1	4.000	66.667	66.667	4.000	66.667	66.667	3.197	53.280	53.280
2	1.577	26.289	92.956	1.577	26.289	92.956	2.381	39.676	92.956
3	.423	7.044	100.000						
4	-1.780E-16	-2.967E-15	100.000						
5	-3.320E-16	-5.534E-15	100.000						
6	-5.233E-16	-8.722E-15	100.000						

Método de Extração: análise de Componente Principal.

Comunalidades

	Inicial	Extração
Completude	1.000	.652
Validade	1.000	.992
Integridade	1.000	.975
Integração	1.000	.992
Reuso	1.000	.975
Minimalisação	1.000	.992

Método de Extração: análise de Componente Principal.

Matriz de componente[a]

	Componente	
	1	2
Completude	.784	-.192
Validade	.906	.414
Integridade	-.679	.717
Integração	.906	.414
Reuso	-.679	.717
Minimalisação	.906	.414

Método de Extração: análise de Componente Principal.
a. 2 componentes extraídos.

Figura 25 – Análise fatorial com indicação de dois componentes para as variáveis observadas na Qualidade Semântica

componente uma vez que o esforço necessário para modificar ou adaptar um meta-modelo para uso em outros contextos variaria dependendo das regras e restrições do domínio. Assim, este critério foi observado a partir dos dois componentes: QS-Conhecimento do Domínio e QS-Aplicabilidade.

A Figura 26 ilustra a análise fatorial para o componente QS-Conhecimento do Domínio. Nesta observa-se que todas as variáveis superaram os mínimos indicados na literatura, apresentando os valores $0,577$ para correlação e $0,5$ e $0,971$ para comunalidades. Ainda, o critério de *Kaiser* sugere que deve-se extrair apenas um fator (componente) que apresenta um *eigenvalue* de 3,414 carregando 85,355% da variância das variáveis originais.

Como mencionado anteriormente, o índice de cada fator (escore) é salvo como uma variável que apresenta um valor para cada caso observado. Para se chegar ao indicador do critério, os escores foram normalizados para uma escala padronizada que vai de 0 a 1. Após a normalização, a média aritmética dos escores foi realizada para se chegar ao valor do indicador. A Figura 27 ilustra a estatística descritiva com o resultado da análise para o indicador de Qualidade Semântica (Conhecimento do Domínio) que possui média de 75%. Este resultado indica um bom nível de Qualidade Semântica no que se refere ao conhecimento do domínio. Considerando que as variáveis apresentaram um ótimo valor

Matriz de correlações[a]

		Completude	Validade	Integração	Minimalisação
Correlação	Completude	1.000	.577	.577	.577
	Validade	.577	1.000	1.000	1.000
	Integração	.577	1.000	1.000	1.000
	Minimalisação	.577	1.000	1.000	1.000

a. Essa matriz não é positiva definida.

Comunalidades

	Inicial	Extração
Completude	1.000	.500
Validade	1.000	.971
Integração	1.000	.971
Minimalisação	1.000	.971

Método de Extração: análise de Componente Principal.

Variância total explicada

	Autovalores iniciais			Somas de extração de carregamentos ao quadrado		
Componente	Total	% de variância	% cumulativa	Total	% de variância	% cumulativa
1	3.414	85.355	85.355	3.414	85.355	85.355
2	.586	14.645	100.000			
3	4.077E-17	1.019E-15	100.000			
4	.000	.000	100.000			

Método de Extração: análise de Componente Principal.

Figura 26 – Análise Fatorial para o componente QS-Conhecimento do Domínio

de correlação e que o fator extraído carrega 85,355% da variância das variáveis.

Estatística Descritiva

	N	Mínimo	Máximo	Média	Desvio
Indice_QS_Dominio	4	.00	1.00	**.7494**	.49960
N válido (de lista)	4				

Figura 27 – Índice de Qualidade Semântica (Conhecimento do Domínio)

A Figura 28 ilustra a análise fatorial para o componente QS-Aplicabilidade. Nesta observa-se que todas as variáveis superaram os mínimos de 0,3 para correlação e 0,4 para comunalidade, apresentando valor 1 para ambos. Observa-se também que o critério de *Kaiser* sugere que deve-se extrair apenas um fator (componente) que apresenta um *eigenvalue* de 2,000 carregando 100% da variância das variáveis originais.

Após execução da análise fatorial e redução dos dados se chegou ao valor de 75% para o indicador de Qualidade Semântica (Aplicabilidade). Este resultado indica um excelente nível de Qualidade Semântica no que se refere a aplicabilidade do meta-modelo. Considerando que as variáveis apresentaram um excelente valor de correlação e que o fator extraído carrega 100% da variância das variáveis. A Figura 29 ilustra a estatística descritiva com o resultado desta análise.

6.1.3.3 Qualidade Organizacional - Q3

Com o indicador de Qualidade Organizacional busca-se identificar se o meta-modelo cumpre os objetivos da modelagem no que se refere as necessidades (normas, deveres e precei-

Matriz de correlações[a]

		Integridade	Reuso
Correlação	Integridade	1.000	1.000
	Reuso	1.000	1.000

a. Essa matriz não é positiva definida.

Comunalidades

	Inicial	Extração
Integridade	1.000	1.000
Reuso	1.000	1.000

Método de Extração: análise de Componente Principal.

Variância total explicada

	Autovalores iniciais			Somas de extração de carregamentos ao quadrado		
Componente	Total	% de variância	% cumulativa	Total	% de variância	% cumulativa
1	2.000	100.000	100.000	2.000	100.000	100.000
2	.000	.000	100.000			

Método de Extração: análise de Componente Principal.

Figura 28 – Análise Fatorial para o componente QS-Aplicabilidade

Estatística Descritiva

	N	Mínimo	Máximo	Média	Desvio
Indice_QS_Aplicabilidade	4	.00	1.00	**.7500**	.50000
N válido (de lista)	4				

Figura 29 – Índice de Qualidade Semântica (Aplicabilidade)

tos) próprios do domínio. Para este foram observados os níveis indicados pelos especialistas para as variáveis Flexibilidade (em termos de extensibilidade e adaptabilidade do meta-modelo), Integração (nos termos da consistência do meta-modelo com as necessidades do domínio) e Implementabilidade (em termos de diversidade de aplicação e custo de implementação). A Figura 22 apresentou as notas indicadas pelos especialistas para as variáveis Flexibilidade, Integração e Implementabilidade. Em todas as variáveis a média das notas foi de $4,75$, indicando um ótimo nível de satisfação para estes critérios do meta-modelo, uma vez que a nota máxima esperada é 5. Para reforçar a validade deste indicador estas notas também foram analisadas estatisticamente seguindo o método de análise fatorial.

As correlações entre as variáveis superaram o mínimo de 0,3 (Figura 30 - Matriz de correlações), e as comunalidades das variáveis também ficaram acima do mínimo de 0,4 (Figura 30 - Comunalidades). As variáveis apresentaram valor 1 para ambos. Logo, os dados são adequados à análise fatorial.

O passo seguinte é determinar o número de fatores (componentes) que serão extraídos. Na Figura 30 (Variância Total Explicada) o critério de *Kaiser* sugere que deve-se extrair apenas um fator (componente) que apresenta um *eigenvalue* de 3,000 carregando 100% da variância das variáveis originais.

Como mencionado anteriormente, o índice de cada fator (escore) é salvo como uma variável que apresenta um valor para cada caso observado. Para se chegar ao indicador do critério, os escores foram normalizados para uma escala padronizada que vai de 0 a

Matriz de correlações[a]

		Flexibilidade	Integração	Implementabbilidade
Correlação	Flexibilidade	1.000	1.000	1.000
	Integração	1.000	1.000	1.000
	Implementabbilidade	1.000	1.000	1.000

a. Essa matriz não é positiva definida.

Comunalidades

	Inicial	Extração
Flexibilidade	1.000	1.000
Integração	1.000	1.000
Implementabbilidade	1.000	1.000

Método de Extração: análise de Componente Principal.

Variância total explicada

	Autovalores iniciais			Somas de extração de carregamentos ao quadrado		
Componente	Total	% de variância	% cumulativa	Total	% de variância	% cumulativa
1	3.000	100.000	100.000	3.000	100.000	100.000
2	-2.220E-16	-7.401E-15	100.000			
3	-4.441E-16	-1.480E-14	100.000			

Método de Extração: análise de Componente Principal.

Figura 30 – Análise fatorial para o componente Qualidade Organizacional

1. Após a normalização, a média aritmética dos escores foi realizada para se chegar ao valor do indicador. A Figura 31 ilustra a estatística descritiva com o resultado da análise para o indicador de Qualidade Organizacional que possui média de 75%. Este resultado indica um excelente nível de Qualidade Organizacional. Considerando que as variáveis apresentaram um excelente valor de correlação e que o fator extraído carrega 100% da variância das variáveis.

Estatística Descritiva

	N	Mínimo	Máximo	Média	Desvio
Indice_Q_Organizacional	4	.00	1.00	**.7500**	.50000
N válido (de lista)	4				

Figura 31 – Índice de Qualidade Organizacional

6.1.3.4 Qualidade Pragmática - Q4

Com o indicador de Qualidade Pragmática busca-se identificar o nível de interpretação do meta-modelo (como este foi entendido pelo leitor). Esta pode se diferenciar entre Qualidade Pragmática Social (no que se refere a compreensão de pessoas) e Qualidade Pragmática Técnica (compreensão de ferramentas). Para este foi observada apenas a Qualidade Pragmática Social através do nível indicado pelos especialistas para a variável Compreensibilidade. A Figura 22 apresentou as notas indicadas pelos especialistas para a variável Compreensibilidade. A média das notas para esta variável foi de $4,75$, indicando um ótimo

nível de satisfação para esta propriedade do meta-modelo, uma vez que a nota máxima esperada é 5. Como este fator apresenta apenas uma variável, o valor desta já foi considerado para representar o indicador, não sendo necessária a redução de dados. Assim, a análise fatorial não foi aplicada para este critério de qualidade.

Para se chegar ao indicador de Qualidade Pragmática os valores das notas foram normalizados para uma escala padronizada (0 a 1) e em seguida foi realizada a média aritmética dos escores obtidos após a normalização. A Figura 32 ilustra a estatística descritiva com o resultado para o indicador de Qualidade Pragmática com média de 75%. Este resultado indica um excelente nível de Qualidade Pragmática.

Estatística Descritiva

	N	Mínimo	Máximo	Média	Desvio
Indice_Q_Pragmatica	4	.00	1.00	**.7500**	.50000
N válido (de lista)	4				

Figura 32 – Índice de Qualidade Pragmática

6.1.3.5 Discussão

Com a execução desta avaliação foram analisadas as notas dos especialistas e extraídos os indicadores de qualidade com a técnica de análise fatorial. Ambos os dados, indicaram que o meta-modelo possui um bom nível de qualidade geral. A Tabela 11 apresenta os resultados finais obtidos para os critérios analisados.

Indicadores de Qualidade		Nota dos Especialistas	Valor dos Indicadores
Qualidade Empírica		4,75	50%
Qualidade Semântica	QS Aplicabilidade	4,75	75%
	QS Conhecimento do domínio	4,7	75%
Qualidade Pragmática		4,75	75%
Qualidade Organizacional		4,75	75%

Tabela 11 – Resultado das análises dos Indicadores de qualidade

Estes apresentaram médias acima de 4,7 (em um máximo de 5) para as notas indicadas pelos especialistas em todos os indicadores de qualidade. Para os indicadores obtidos com a Análise fatorial estes apresentaram uma maioria com média em 75% e apenas um indicador apresentou sua média em 50%.

A **Qualidade Sintática** tem o sentido de corretude sintática e se refere ao uso correto do vocabulário da linguagem utilizada para desenvolvimento do meta-modelo. Assim, este indicador já é auto verificado com o uso das ferramentas de modelagem para criação de modelos com o Ecore EMF. Assim, não foi necessário observar e analisar estatisticamente os critérios relacionados a este indicador de Qualidade Sintática.

Com esta avaliação ficou comprovada a adequação do método estatístico de análise fatorial para geração destes indicadores de qualidade. É importante mencionar que um esforço para validar a relevância dos conjuntos de critérios observados em cada indicador ainda se faz necessário pois observou-se, por exemplo, que o componente que apresentou os menores valores para correlação (0,333) entre as variáveis observadas foi o que teve seu indicador em 50%. Como este método de análise se baseia no padrão de correlação observado entre as variáveis, indicadores que analisam variáveis com maiores valores de correlação se adéquam melhor ao método mas é importante considerar que este fato também pode estar relacionado com aos critérios observados e o número reduzido de casos analisados. Assim, se faz necessário a aplicação do método em um contexto que possibilite a coleta de um volume maior de dados, não só para validar a aplicação do método, mas também para coletar evidências de que este pode ser generalizado para variados contextos.

6.2 OBSERVAÇÃO PARTICIPANTE

O objetivo deste estudo etnográfico foi coletar informações sobre uso e compreensão da linguagem de modelagem GenML. A observação foi realizada com o intuito de capturar comportamentos e interações dos usuários com a linguagem e ferramenta de modelagem desenvolvida. Para este estudo foi proposto um exercício a ser realizado com o auxílio da ferramenta de modelagem.

6.2.1 Planejamento e Execução do Estudo

O exercício proposto para este estudo está disponível no Apêndice B. Esta atividade propôs que os usuários realizassem a modelagem de uma fonte de dados utilizando dois dos diagramas disponíveis na ferramenta (o diagrama da fonte e o diagrama de entidade). O experimento foi realizado individualmente e usuários utilizaram a mesma máquina para o exercício.

6.2.1.1 Contexto e Variáveis Medidas

As variáveis observadas estão relacionadas ao uso da linguagem e ferramenta de modelagem.

- *TM* - Tempo de Modelagem: quantidade de minutos gastos para executar a tarefa de modelagem.
- *EC* - Erros de Compreensão: quantidade de elementos mal compreendidos/interpretados pelos usuários.

- *CO* - Consultas ao Observador: quantidade de vezes que o observador foi consultado para solução de dúvidas.
- *DNV* - Diagrama Não Validado: diagrama não aprovados pela validação da ferramenta.
- *EA* - Erros de Aplicação: quantidade de elementos aplicados de maneira incorreta no modelo. Erros identificados pela validação da ferramenta.
- *DI* - Diagramas Incompletos: quantidade de elementos com requisitos (atributos ou sub-elementos) obrigatórios não atendidos pelo modelo.
- *EF* - Erros da Ferramenta: quantidade de erros permitidos pela ferramenta. Erros não identificados pela validação da ferramenta.
- *PU* - Problemas de Usabilidade: problemas de uso e manuseio dos elementos da interface gráfica.

6.2.1.2 Coleta de Dados: Ficha e Observações

A coleta de dados é sempre realizada em relação a uma unidade de análise bem definida. Nesta avaliação a unidade de análise será composta da *persona*: Desenvolvedor. Um Desenvolvedor é considerado neste trabalho como um profissional de computação com atuação no desenvolvimento de sistemas tendo conhecimentos básicos de modelagem de software e bancos de dados. Para esta *persona* contamos com a participação de cinco desenvolvedores com níveis de experiência variados nos tópicos solicitados.

Esta avaliação fez uso da técnicas de coleta de dados observação. Esta técnica foi escolhidas por permitir a coleta de dados tanto qualitativos quanto quantitativos além de se apresentar como instrumento tradicional para o tipo de coleta de dados que se pretende realizar. A observação participante é o estudo etnográfico mais adequado para coletar informações sobre uso e compreensão de ferramentas sem interferência direta do observador e sem que o usuário informe explicitamente a sua opinião sobre o elemento em estudo (EASTERBROOK et al., 2007).

Os dados foram coletados pelo pesquisador observador por meio do preenchimento de uma ficha de observação onde foram realizadas anotações sobre a ocorrência das variáveis e outras observações que pudessem ser interessantes para a o objetivo desta avaliação. A coleta dos dados desta técnica foi realizada sistematicamente e de forma não intrusiva. A ficha de observação utilizada para coleta dos dados está disponível no Apêndice C.

6.2.1.3 Dados Coletados

Como mencionado anteriormente, os dados foram coletados pelo pesquisador observador por meio do preenchimento de uma ficha de observação. Os dados coletados representam

a frequência da ocorrência de um conjunto de variáveis relacionadas ao uso da ferramenta e compreensão da linguagem de modelagem GenML. A Figura 33 ilustra a frequência das variáveis que tiveram sua ocorrência observadas neste estudo.

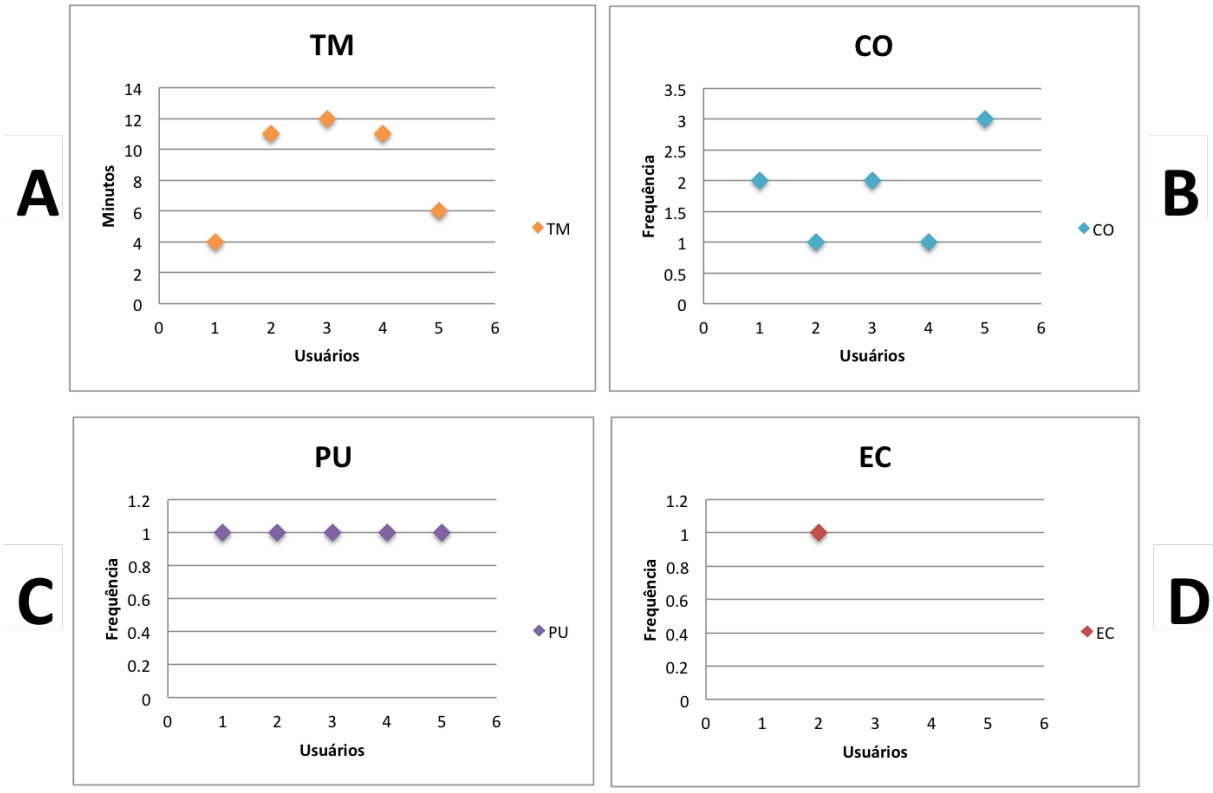

Figura 33 – Gráficos com frequência das variáveis TM, CO, PU e EC por usuário

6.2.2 Análises e Interpretação dos Resultados Baseado no Método GQM

O objetivo deste estudo foi definido como segue:

Objetivo: Analisar a ferramenta e linguagem GenML quanto à avaliação de seu uso e compreensão do ponto de vista dos desenvolvedores no contexto de modelagem de bancos de dados de genoma.

Depois de definir o objetivo, ele foi então refinado em questões que tentam caracterizar os objetos de medição. Nesta análise foram definidas as questões a seguir:

Q1: *Quais os problemas de usabilidade da ferramenta de modelagem de acordo a observação das variáveis CO, PU e EF?*

Q2: *Quão compreensível é a linguagem GenML de acordo as ocorrências das variáveis EC, DNV e EA?*

A análise dos dados coletados foi iniciada pelo perfil dos usuários. Todos eles indicaram ter conhecimento médio em modelagem de software e banco de dados. Apenas dois

indicaram ter experiência profissional em atividades de modelagem, os demais só tiveram contato direto com modelos durante os períodos de formação acadêmica. Nenhum dos participantes relatou ter conhecimento alto em modelagem de software ou banco de dados. Devido à homogeneidade dos participantes (grande maioria com conhecimento médio), esta informação não foi utilizada para correlação com os resultados das variáveis observadas.

Uma das variáveis observadas foi o tempo gasto para modelar os diagramas(variável TM). A informação de horário de início e término para elaborar cada diagrama foi anotada pelo observador no formulário de observação. A Figura 33A ilustra o tempo gasto por cada usuário para conclusão da modelagem. A média do tempo ($avg(TM_1, ..., TM_5)$) gasto para conclusão da modelagem foi de $9 minutos$.

A quantidade de vezes que os usuários acionaram o observador para sanar dúvidas sobre a ferramenta também foi observada (variável CO). Todos os usuários consultaram o observador pelo menos uma vez. As dúvidas dos usuários foram provocadas principalmente por insegurança no uso de alguns botões para modificações do diagrama (exclusões, por exemplo) e problemas de usabilidade da ferramenta. Os usuários fizeram perguntas como "*onde alterar o nome de um elemento?*", "*como alterar o tipo de dado de um atributo?*","*como excluir elementos do diagrama?*". A Figura 33B ilustra a frequência da variável CO.

Como mencionado anteriormente, a ferramenta apresentou alguns problemas de usabilidade (variável PU) que devem ser levados em consideração para melhorar a experiência do usuário. Todas as ocorrências desta natureza foram devidamente anotadas pelo observador e serão consideradas para melhoria da ferramenta. A Figura 33C ilustra a frequência da variável PU.

Assim, concluiu-se que devem ser realizadas algumas melhorias na usabilidade da ferramenta para melhorar a experiência do usuário. Como exemplo de melhorias, será necessário adicionar algumas facilidades como a alteração de nomes e exclusão de elementos diretamente no diagrama.

Os diagramas desenvolvidos pelos participantes do experimento foram avaliados com base nos requisitos básicos que deveriam ser alcançados para categorizar se os diagramas estariam ou não completos. Para esta avaliação foram verificados tanto a utilização dos elementos obrigatórios do meta-modelo quanto os atributos definidos na estrutura da entidade proposta no exercício. Nenhum dos usuários concluiu a atividade com diagramas incompletos (variável DI).

Outra variável observada nesse experimento foi a quantidade de componentes utilizados incorretamente na elaboração dos diagramas (variável EC, Figura 33D). Esta variável se preocupou em observar erros de compreensão do usuário na utilização dos elementos gráficos da linguagem. Apenas um usuário cometeu este tipo de erro confundindo o elemento gráfico de entidade com o de atributo. O usuário percebeu a falha e corrigiu o

diagrama sem intervenção do observador.

Os diagramas foram avaliados ainda quanto ao uso correto dos elementos para categorizar se os diagramas estariam ou não válidos (aprovados pela validação da própria ferramenta). A validação da ferramenta verifica se os modelos estão em conformidade com as regras do meta-modelo (sintaxe abstrata da linguagem de modelagem). Nenhum dos diagramas apresentou este tipo de problema (variável DNV) mas é importante mencionar que a ferramenta foi desenvolvida com recursos que impedem estes tipos de erros. Não é possível que o usuário crie uma versão de uma fonte (*SourceVersion*) sem antes ter criado o elemento *Source* que irá recebê-lo, por exemplo. Assim, os usuários são levados a obedecer as regras e hierarquia definidas pelo meta-modelo. Desta maneira, a ferramenta também contribuiu para que não houvesse a ocorrência de casos para as variáveis EA e EF.

Para responder a questão $Q1$ de uma maneira quantitativa foi definida uma equação com base na média aritmética da ocorrência das variáveis (métricas) relacionadas a usabilidade da ferramenta. Esta equação foi definida como segue:

$$U = \frac{CO+PU+EF}{NP}$$

onde:

- U é o indicador de usabilidade;
- CO é o número de ocorrências da variável EC;
- PU é o número de ocorrências da variável DNV;
- EF é o número de ocorrências da variável EA; e
- NP é o número de participantes do estudo.

Para responder a questão $Q2$ de uma maneira quantitativa foi definida uma equação com base na média aritmética da ocorrência das variáveis (métricas) relacionadas a compreensão do modelo. Esta foi definida como segue:

$$C = \frac{EC+DNV+EA}{NP}$$

onde:

- C é o indicador de compreensão;
- EC é o número de ocorrências da variável EC;
- DNV é o número de ocorrências da variável DNV;
- EA é o número de ocorrências da variável EA; e
- NP é o número de participantes do estudo.

6.2.2.1 Discussão

Após calcular os resultados, foi obtido como resposta para a questão $Q1$ o valor $2,8$, indicando que melhorias de usabilidade ainda devem ser realizadas na ferramenta de modelagem para melhorar a experiência do usuário uma vez que este valor representa que 56% dos usuários enfrentaram algum problema de usabilidade.

Já para a questão $Q2$, foi obtido o valor $0,2$ indicando que a linguagem GenML possui um ótimo nível de compreensão uma vez que este valor representa que apenas 4% dos usuários demonstraram não terem compreendido algum elemento da linguagem GenML. Mas é importante mencionar ainda, que este problema de compreensão foi resolvido pelo próprio usuário sem intervenção do observador e o exercício foi concluído com sucesso.

6.3 LIMITAÇÕES, VALIDADE E CONFIABILIDADE DAS AVALIAÇÕES

A validade de um estudo denota a confiabilidade dos resultados, isto é, em que medida os resultados são verdadeiros e não enviesados pelo ponto de vista subjetivo dos pesquisadores. Assim, é importante escolher uma unidade apropriada de análise para garantir que o estudo se concentre nos fenômenos pretendidos. A principal fraqueza dos estudos de caso é que a coleta e a análise de dados estão mais abertas à interpretação e ao viés do pesquisador. Por esse motivo, uma estrutura explícita é necessária para selecionar casos e coletar dados. Embora um estudo de caso individual revele *insights* profundos, a validade dos resultados depende de uma estrutura mais ampla de indução empírica (YIN, 2008; RUNESON; HÖST, 2009).

Este estudo optou por usar um questionário que coletou dados quantitativos (escala likert com níveis de satisfação indicada pelos especialistas) e qualitativos (respostas abertas). Durante o estudo, também foram realizadas observações a usuários e suas interações com a ferramenta de modelagem para identificar possíveis indícios de incompreensão ou mal funcionamento da plataforma além de verificar quanto tempo levou para os usuários concluírem as tarefas definidas para eles.

Assim, as ameaças à validade são divididas em quatro classes: (1) Interna, (2) Externa, (3) Construção e (4) Confiabilidade (RUNESON; HÖST, 2009). A seguir serão listadas as ameaças identificadas e algumas ações para mitigá-las.

6.3.1 Validade de Construção

Seleção dos participantes: em estudos que envolvem a participação humana os resultados dependem das pessoas entrevistadas e a falta de experiência pode levar a equívocos. Com o objetivo de mitigar essa ameaça, selecionamos 4 especialistas do domínio com formações diversas e diferentes níveis de experiência e 5 desenvolvedores também com

níveis de experiência diferenciados. Portanto, a solução aqui proposta foi avaliada por diversas perspectivas.

Viés de reação: existe o risco de que a presença do investigador do estudo influencie seu resultado. Assim, obtendo resultados favoráveis devido ao comportamento tendencioso. Para reduzir esse problema, em uma das técnicas o pesquisador esteve presente apenas durante o treinamento dos participantes e a coleta dos dados foi realizada via formulário online. Na atividade de observação é necessário que o pesquisador esteja presente mas este observou a execução da tarefa sem comentar ou interagir com nenhum participante durante a execução do experimento, apenas observando e anotando discretamente. As interações só ocorreram quando solicitadas pelo participante. É importante mencionar que todos os participantes receberam o mesmo treinamento para uma introdução ao metamodelo e ao propósito da pesquisa e das ferramentas que compõem a arquitetura domínio. Mas é importante mencionar também que este treinamento não deu detalhes sobre o uso da ferramenta e nem outras informações que pudessem enviesar as avaliações. O treinamento durou em média 52 minutos e o material disponibilizado para os usuários nesta ocasião está disponível no Apêndice D.

Pontualidade do questionário: Um desafio é garantir que as perguntas sejam projetadas de maneira a produzir dados úteis e válidos. Pode ser difícil formular as perguntas de modo que todos os participantes as compreendam da mesma forma. Assim, perguntas podem ser mal entendidas ou respostas mal interpretadas, resultando em respostas erradas. Para evitar isso, o experimentador teve que validar as respostas, fazendo as correções que achasse necessário.

6.3.2 Ameaça Interna

A validade interna, ou credibilidade, está relacionada à medida em que os resultados correspondem à realidade e que os pesquisadores foram capazes de capturar a realidade o mais próximo possível. Neste trabalho os dados foram coletados diretamente de especialistas do domínio via questionário. Em seguida, os dados foram contrastados e comparados com os dados coletados com observações do observador-participante. Ainda, foi utilizada uma análise estatística para processar os dados e gerar indicadores de qualidade. Esta verificação estatística indicou uma boa correlação entre as variáveis observadas mas é importante reconhecer que a quantidade de casos analisados não é estatisticamente significante pois, devido a especificidade do domínio, não foi possível encontrar um número maior de especialistas.

6.3.3 Ameaça Externa

Pesquisador específico: Este estudo ocorreu com poucos especialistas devido a dificuldade de encontrar outros profissionais que se encaixassem nos requisitos de seleção. No entanto,

para minimizar essa ameaça, o contexto foi detalhado para facilitar a compreensão de como os resultados encontrados aqui podem se aplicar a outro contexto específico. Além disso, dados de várias fontes foram usados para validar a opinião do pesquisador. No entanto, a medida que os resultados do estudo podem ser generalizados para outros contextos e para outras pessoas ainda é um problema, principalmente devido a insignificância estatística dos casos observados.

6.3.4 Confiabilidade

Interpretação dos dados: o resultado pode ser influenciado pela interpretação dos dados pelo pesquisador do estudo, inserindo um viés de interpretação. Para atenuar essa ameaça, o estudo foi projetado para coletar dados de várias fontes e possibilitar a triangulação desses dados.

6.4 CONSIDERAÇÕES

Este capítulo apresentou a avaliação das abordagens propostas neste trabalho. Para esta avaliação, foram realizados estudos que coletaram dados qualitativos e quantitativos por meio de questionário com especialistas e observação de usuários.

Com a avaliação por meio de questionário com especialistas, foi possível chegar aos indicadores de qualidade do meta-modelo genDB. Os resultados desta avaliação demonstraram alta satisfação dos especialistas quanto ao meta-modelo, demonstrando que este é capaz de representar os dados biológicos moleculares.

Com a observação participante, foi possível verificar a usabilidade da ferramenta CASE e a compreensibilidade da linguagem GenML. Este estudo indicou que algumas melhorias de usabilidade devem ser adicionadas a ferramenta CASE para melhorar a experiência do usuário e confirmou também que a linguagem GenML é facilmente compreensível pelos usuários.

Estes estudos confirmaram também a relevância e aplicabilidade das soluções no processo de anotação de variantes genéticas pois, com a ajuda das ferramentas desenvolvidas, a abordagem se demonstrou capaz de auxiliar os desenvolvedores e bioinformatas na preparação dos dados para realização de anotação de variantes.

Adicionalmente o questionário com especialistas também contribuiu para aplicação da abordagem proposta para avaliação de modelos conceituais. Com a execução da avaliação, foi comprovada a adequação do método estatístico de análise fatorial para geração dos indicadores de qualidade. É importante mencionar que o processo metodológico para engenharia de domínio e os guias operacionais de avaliação de modelos conceituais foram validados pela aplicação neste trabalho mas ainda é necessário aplicá-los em outros projetos para coletar evidências de que estes podem ser generalizados para variados contextos.

É importante mencionar que após a implementação também foram realizados experimentos com o algoritmo de identificação de esquemas. Estes experimentos tiveram o objetivo de exercitar e demonstrar o desempenho do algoritmo, coletando dados da execução da ferramenta em diferentes cenários. Este experimento demonstrou que ainda existem limitações e que melhorias de desempenho devem ser realizadas. Como o tratamento de grandes volumes de dados não é um objetivo desta pesquisa, as questões de desempenho serão consideradas nos trabalhos futuros.

No Capítulo 7 será apresentado esta e outras limitações que podem ser abordadas nos trabalhos futuros a serem realizados a partir desta pesquisa. Também serão apresentadas as considerações finais destacando como os objetivos da pesquisa foram atendidos.

7 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

"Um poema nunca se acaba; apenas se abandona."

—Paul Valery

Este capítulo apresenta as considerações finais desta pesquisa destacando os principais benefícios e contribuições da abordagem proposta. São discutidos os objetivos alcançados e as contribuições relacionadas a estes objetivos. Também serão apresentadas as limitações e os trabalhos futuros que podem ser desenvolvidos a partir desta pesquisa. Ao final do capítulo serão listadas as publicações originadas deste trabalho.

7.1 ATENDIMENTO AOS OBJETIVOS

No Capítulo 1 foi apresentado o objetivo principal desse trabalho que se destinou a especificar um meta-modelo para representação do conjunto de dados envolvido no processo de análise de dados biológicos moleculares (especificamente na etapa de anotação de variantes de genoma). Este meta-modelo possibilitou a implementação de ferramentas de apoio aos bioinformatas na preparação dos dados para anotação de variantes genéticas, auxiliando na identificação e visualização dos esquemas dos dados.

A partir do objetivo geral, seis objetivos específicos foram definidos. Em seguida, será discutido como estes objetivos foram alcançados.

7.1.0.1 OE1 - Analisar o domínio de dados biológicos moleculares.

Nesse trabalho, uma das contribuições foi o resultado da análise do domínio de dados biológicos moleculares que identificou os principais conceitos e características dos dados envolvidos no processo de anotação de variantes de genoma. Essa análise resultou não só no mapeamento dos conceitos como também na categorização dos tipos de esquemas presentes nos conjuntos de dados biológicos moleculares. Os conjuntos de dados analisados são disponibilizados periodicamente por diversos publicadores (fontes) no formato XML. Este formato de dado permite a flexibilidade das estruturas nas quais os dados são publicados, levando a uma grande variação nos esquemas destes dados, o que se torna um desafio na manipulação destes conjuntos de dados. Esta natureza flexível das bases de dados levou a necessidade do conceito de esquemas versionados. A noção de versão permitiu gerenciar os conjuntos de objetos de mesmo rótulo, que têm diferentes esquemas e, consequentemente, auxiliar o entendimento e manipulação destes.

Após a identificação dos conceitos e seus relacionamentos, foi possível entender o comportamento das estruturas e chegar a definição de três tipos de esquemas: (1) O Esquema da Versão da Fonte, com as representações das diversas versões de suas entidades; (2) O Esquema da Fonte, com a união dos esquemas de todas as suas versões; e (2) O Esquema da Entidade, com a junção de todos os atributos presentes nas entidades de mesmo tipo. Estes esquemas foram definidos com o intuito de possibilitar não só o entendimento e visualização das estruturas mas também o acompanhamento da evolução destes esquemas.

Conhecendo os conceitos e os diversos tipos de esquemas, foi possível modelar o domínio e projetar as ferramentas para apoio aos bioinformatas na preparação dos dados para anotação de variantes genéticas. Estes conceitos e tipos de esquemas foram apresentados na Seção 4.1 do Capítulo 4.

7.1.0.2 OE2 - Especificar um meta-modelo que explique/represente o conjunto de dados biológicos moleculares.

A principal contribuição desse trabalho foi a especificação do meta-modelo GenDB para representação de dados biológicos moleculares. Este meta-modelo apresentou os conceitos do domínio e as correspondências e relacionamentos entre estes conceitos. Entre os principais conceitos que compõem o meta-modelo GenDB estão: *Source*, *SourceVersion*, *Entity*, *Attributes*, *Value*, e *Association*. Com estes conceitos, o meta-modelo GenDB suporta a representação dos conjuntos de dados biológicos moleculares sob os três tipos de esquemas definidos para o domínio. A definição do meta-modelo GenDB foi apresentada na Seção 4.2 do Capítulo 4.

O meta-modelo GenDB também permitiu projetar e implementar ferramentas para apoio aos bioinformatas na preparação dos dados para anotação de variantes genéticas. Este meta-modelo foi validado com a implementação destas ferramentas, que foram apresentadas no Capítulo 5. Com a implementação foi possível demonstrar que o meta-modelo não só é capaz de representar os conjuntos de dados biológicos mas também permitiu a criação de ferramentas para visualização e manipulação dos esquemas de dados biológicos moleculares, facilitando o entendimento da estrutura dos dados e, consequentemente, auxiliando os bioinformatas. A qualidade do meta-modelo GenDB foi verificada por meio de um estudo baseado na opinião de especialistas. Este estudo foi apresentado no Capítulo 6 e demonstrou que o meta-modelo tem altos indicadores de qualidade além de possuir um ótimo nível de satisfação dos especialistas. Os indicadores de qualidade verificados foram: Qualidade Empírica; Qualidade Semântica; Qualidade Pragmática; e Qualidade Organizacional.

É importante mencionar que meta-modelo GenDB foi validado para aplicação em bases de dados semi-estruturadas XML. Assim, ainda se faz necessária a aplicação deste para

bases de dados sob outros formatos (VCF, por exemplo). Esta limitação será abordada como um dos trabalhos futuros dessa tese.

7.1.0.3 OE3 - Especificar uma linguagem de modelagem específica de domínio.

Nesse trabalho, uma outra contribuição foi a linguagem de modelagem GenML que possui uma notação visual e apresenta diversos diagramas para visualização dos diferentes tipos de esquemas definidos para bancos de dados biológicos moleculares. Para a criação desta linguagem, primeiramente foi definida a sintaxe concreta do meta-modelo GenDB, seguindo os princípios de Moody para construção de notações visuais. Em seguida, esta linguagem de modelagem foi validada com a implementação de uma ferramenta CASE (GenModelCASE). A ferramenta GenModelCASE foi implementada usando os recursos oferecidos pela Sirius para criação de ferramentas de modelagem. A definição da linguagem GenML foi apresentada na Seção 4.3 do Capítulo 4 e a implementação da ferramenta GenModelCASE foi apresentada no Capítulo 5.

A linguagem GenML contribuiu para validar o meta-modelo GenDB na representação de dados biológicos moleculares. Esta linguagem foi avaliada com apoio da ferramenta GenModelCASE por meio de um estudo etnográfico com desenvolvedores. Este estudo realizou uma observação participante para coleta de dados sobre a compreensão da linguagem GenML e o uso da ferramenta GenModelCASE. A avaliação identificou que a linguagem GenML apresenta uma ótima compreensibilidade mas que melhorias de usabilidade ainda devem ser implementadas na ferramenta CAGenModelCASE, com o intuito de melhorar a experiência do usuário. Uma das limitações identificadas na ferramenta GenModelCASE foi na visualização de esquemas com muitas versões de entidades de mesmo tipo, onde recursos como filtros e destaques podem contribuir para melhorar a manipulação dos modelos. A avaliação da linguagem GenML foi apresentada no Capítulo 6.

7.1.0.4 OE4 - Especificar uma abordagem de identificação de esquema.

O algoritmo de engenharia reversa baseada em modelos para identificar os esquemas implícitos em bancos de dados semi-estruturados XML, se apresenta como uma outra contribuição desse trabalho. A estratégia definida neste algoritmo levou em conta os conceitos presentes nas bases de dados biológicos moleculares e no meta-modelo GenDB. Assim, os esquemas identificados nos conjuntos de dados podem ser transformados em modelos que estão de acordo com o meta-modelo GenDB. A definição deste algoritmo foi apresentada na Seção 4.4.1 do Capítulo 4.

Este algoritmo facilitou a manipulação e entendimento dos esquemas pois, além de identificar as estruturas presentes nas bases de dados, esta solução foi implementada

com base na transformação entre modelos (T2M) e permite a geração de um modelo *.gendb*, que pode ser visualizado com a linguagem GenML. Este algoritmo foi validado pela sua implementação e aplicação em duas bases de dados(*ClinVar* e *GenBank*). A implementação deste foi apresentada no Capítulo 5. Após a implementação do algoritmo, também foram realizados experimentos para exercitar e demonstrar a execução deste. Este experimento demonstrou que melhorias de desempenho devem ser realizadas mas, como o tratamento de grandes volumes de dados não é um objetivo desta pesquisa, estas questões de desempenho serão consideradas nos trabalhos futuros.

7.1.0.5 OE5 - Especificar uma abordagem para criação de bases de dados.

Nesse trabalho também foi especificado um algoritmo para geração de esquemas de bases de dados orientadas a documento. Neste algoritmo, os esquemas são gerados a partir dos modelos *.gendb*, considerando os conceitos presentes nas bases de dados biológicos moleculares e no meta-modelo GenDB. A definição deste algoritmo foi apresentada na Seção 4.4.2 do Capítulo 4.

Este algoritmo validou a aplicação do meta-modelo GenDB para a geração de esquemas de bases de dados orientadas a documento, além de auxiliar os desenvolvedores na criação destas bases. O algoritmo foi implementado com base na transformação de modelos (M2T) para geração de esquemas Mongoose de bases MongoDB. A implementação deste algoritmo foi apresentada no Capítulo 5.

7.1.0.6 OE6 - Implementar e Avaliar as soluções apresentadas neste trabalho

Como parte do último objetivo, esse trabalho implementou as ferramentas necessárias para validação das soluções que compõem a abordagem proposta no escopo desse trabalho. Esta implementação foi apresentada no Capítulo 5 e permitiu validar: (1) o meta-modelo GenDB, com sua aplicação na implementação das diversas ferramentas que compõem a arquitetura de referência da plataforma do domínio; (2) a linguagem de Modelagem GenML, com sua aplicação na ferramenta GenModelCASE para visualização e manipulação de modelos *.gendb*; (3) o algoritmo de identificação de esquemas, com sua aplicação em duas fontes de dados XML (*ClinVar* e *GenBank*); (4) o algoritmo para criação de bases de dados, com sua aplicação na geração de esquemas de bases de dados orientadas a documento MongoDB; e (5) a ferramenta cliente (ClinGen) e a API, implementadas e aplicadas para validação das demais camadas da arquitetura definida para o domínio.

Esta implementação contribuiu também para a validação do uso da abordagem de MDE no desenvolvimento de soluções no contexto de engenharia de dados. As principais vantagens e limitações do uso desta abordagem foram discutidas nas considerações do Capítulo 5.

Verificar a qualidade das soluções e analisar empiricamente os resultados obtidos, também comporam o último objetivo desse trabalho. No Capítulo 6, foram apresentados os resultados das avaliações realizadas para verificar a qualidade do meta-modelo GenDB, o uso da ferramenta GenModelCASE, e a compreensão da linguagem de modelagem GenML. As avaliações demonstraram que o meta-modelo GenDB possui uma boa qualidade e que a linguagem de modelagem GenML é bastante compreensível. Para a ferramenta GenModelCASE, a avaliação demonstrou que ainda são necessárias algumas melhorias na usabilidade.

Os testes de execução do algoritmo de identificação de esquemas demonstraram que melhorias de desempenho devem ser realizadas, mas o desempenho não foi o foco principal desse trabalho sendo esta implementação considerada suficiente para validação da abordagem proposta no escopo dessa pesquisa.

A verificação da qualidade do meta-modelo GenDB contribuiu também para a validação da abordagem definida para avaliação de modelos conceituais, demonstrando como conduzir a avaliação e como analisar seus resultados, confirmando também a adequação do método estatístico de análise fatorial para geração dos indicadores de qualidade.

7.2 CONTRIBUIÇÕES

A principal contribuição deste trabalho foi a definição de um meta-modelo capaz de representar os esquemas de bases de dados biológicos moleculares. Este meta-modelo foi desenvolvido com técnicas de MDE, o que possibilitou a criação de ferramentas para apoio ao gerenciamento de esquemas dos dados envolvidos no processo de anotação de variantes de genoma. Entre estas ferramentas pode-se destacar a DSML para visualização dos esquemas versionados das bases de dados biológicos moleculares. Esta linguagem de modelagem também foi definida no escopo desse trabalho.

Esse trabalho apresentou também uma abordagem de engenharia reversa baseada na transformação de modelos para identificar os esquemas das bases, considerando o versionamento das entidades e das fontes de dados. A transformação de modelos também foi aplicada para definir um algoritmo de geração de esquemas para bases de dados orientadas a documento. Todas as soluções propostas e implementadas nesse trabalho contribuíram também para a definição de uma arquitetura de referência para suporte a análise de dados biológicos moleculares.

As principais contribuições desse trabalho podem ser resumidas em:

- A análise realizada para identificação dos conceitos e tipos de esquema presentes nos bancos de dados biológicos moleculares, relacionada ao objetivo OE1;
- O meta-modelo para representação de bases de dados biológicos moleculares, relacionado ao objetivo OE2;

- A linguagem de modelagem para visualização e diagramação dos esquemas das bases de dados biológicos moleculares, reacionada ao objetivo OE3;
- O algoritmo para identificação de esquemas em bases de dados biológicos moleculares, relacionado ao objetivo OE4;
- O algoritmo para geração de esquemas de bases de dados orientadas a documento, relacionado ao objetivo OE5; e
- A aplicação da abordagem de MDE para o desenvolvimento de soluções no contexto de engenharia de dados; relacionada ao objetivo OE6;

Além das contribuições principais, esse trabalho também apresentou uma contribuição secundária:

- A definição de um processo para guiar a engenharia de domínio e a avaliação de qualidade de modelos se caracterizam como uma contribuição secundária desse trabalho pois, além do conjunto de atividades para realizar análise, projeto e implementação do domínio, este processo adicionou um conjunto de diretrizes para avaliação de qualidade. Estas diretrizes apresentaram não só a definição mas também a operacionalização dos critérios a serem observados para geração de indicadores de qualidade estatisticamente validados e calculados.

7.3 LIMITAÇÕES E TRABALHOS FUTUROS

A pesquisa apresentada neste trabalho permitiu que o Grupo de pesquisa ASSERT Lab, o Centro de Informática da UFPE, o Departamento de Computação da UFRPE e o laboratório Genomika iniciassem uma linha de pesquisa em *Model-Driven Data Engineering*, na qual pretende-se colaborar nos próximos anos. A abordagem proposta neste trabalho foi aplicada para esquemas de bases de dados de genoma no formato XML e, no entanto, não se pode afirmar que esta é aplicável em todas as circunstâncias e em todos os contextos de dados semi-estruturados e orientados a agregados. Mais estudos devem ser realizados, com diferentes domínios, a fim de complementar a abordagem, resultados e produtos de trabalho ainda não especificados.

Diversas orientações podem ser tomadas para continuar a atividade de pesquisa nessa área. Assim, os trabalhos futuros que podem ser desenvolvidos com base nesta pesquisa serão caracterizados a seguir.

Aplicação do Meta-modelo

Pretende-se aplicar a solução para outros sistemas e formatos de dados (VCF, por exemplo) existentes no contexto de análise de dados de genoma para verificar o grau de reuso

e implementabilidade do meta-modelo. Como exemplo, pode-se citar os sistemas de gestão de regras de laudos. Este sistema da suporte a regras complexas por laudo, uma vez que estas regras e seus campos variam ao longo do tempo. Pretende-se aplicar a solução também para outros modelos de dados para verificar a generalidade do meta-modelo em relação a outros bancos de dados NoSQL e Relacionais.

Geração de Esquemas de Bancos de Dados

Projeto e implementação de soluções para geração de esquemas e validadores de dados. Para esta atividade pretende-se desenvolver uma solução MDE para gerar os códigos dos esquemas e dos validadores a serem usados para verificar se os dados estão armazenados sem violar as definições do esquema.

Diferenciação de Versões e Evolução

Definir uma estratégia para classificar instâncias. Definir um algoritmo de agrupamento para determinar a qual versão de entidade uma instância pertence. Essa classificação pode ser útil, por exemplo, para criar procedimentos de migração e também para acompanhar a evolução dos dados. Para esta atividade pretende-se pesquisar a aplicação de bases de dados em grafo como um padrão para abordar a heterogeneidade dos dados. Esta atividade também deve prever estratégias de visualização para comparação entre esquemas e acompanhamento da evolução dos dados.

Visualização de Esquemas

Quando muitos esquemas são inferidos, a visualização de todos os esquemas não é facilmente gerenciável. Assim, pretende-se prover funcionalidades de consulta ou outros mecanismos de navegação para melhor entendimento dos esquemas inferidos. Para esta atividade pretende-se criar uma nova versão do editor de esquemas desenvolvido com o Sirius adicionando esses recursos.

Automatização do *Workflow*

Projetar e implementar os elementos computacionais necessários para automatizar todo o fluxo de dados da plataforma evitando assim a necessidade de muitas interações humanas durante o processo de inferência e migração dos dados.

Otimização e Desempenho

Projetar e implementar soluções que melhorem o desempenho dos sistemas desenvolvidos. A ferramenta projetada para inferência de esquemas não apresentou um bom desempenho em grandes volumes de dados. Como o ambiente do domínio estudado é caracterizado por manipular grandes volumes de dados, ainda é necessário pesquisar soluções que melhorem o desempenho do mecanismo de inferência de esquemas em bases de dados volumosas.

Esta atividade irá investigar a utilização do *framework MapReduce* em um *cluster Hadoop*[1] para inferência de esquemas versionados.

Migração de Dados para Persistência Poliglota
Soluções para persistência poliglota tem sido amplamente utilizadas com o surgimento dos mais variados modelos para persistência de dados. Como as arquiteturas de banco de dados NoSQL possuem benefícios e vantagens específicas, é preciso usar vários destes sistemas em uma mesma solução, levando à persistência poliglota. Assim será necessário pesquisar a migração de aplicativos legados para novos sistemas baseados na persistência poliglota. Para esta atividade pretende-se aplicar o meta-modelo proposto para soluções de persistência poliglota.

Qualidade de Modelos Conceituais
A avaliação de modelos conceituais é um tema importante não só no contexto de MDE mas também na engenharia de software e bancos de dados. Este trabalho apresentou uma definição preliminar para avaliação de meta-modelos que pode ser refinado ou estendido para modelos conceituais de outros contextos. Esta atividade pretende pesquisar e analisar as propriedades específicas de cada critério de qualidade proposto e identificar a relevância destes para o critério em análise. Esta relevância pode ser identificada em forma de pesos que serão utilizados de acordo com o contexto de aplicação da avaliação. Para esta atividade deverá ser considerado o conhecimento de especialistas (pesquisadores e profissionais). Um estudo de caso que permita a participação de uma população estatisticamente significante também deve ser selecionado.

7.4 PUBLICAÇÕES

Esta pesquisa produziu várias contribuições que foram apresentadas e discutidas no Capítulo 4. Os artigos publicados com os resultados desta pesquisa estão listados abaixo.

- Este artigo apresentou uma versão preliminar da arquitetura proposta para análise de dados biológicos moleculares e foi publicado em forma de resumo no I *Database Systems Industry Day Workshop* (DSIDW) que ocorreu junto com o Simpósio Brasileiro de Banco de Dados. ALENCAR, A. L.; BURÉGIO, V.; FREITAS CARACIOLO, M. J.; GARCIA, V. A centralized platform for access of heterogeneous data on human genome repositories for supporting clinical decisions. In: Proceedings of the Satellite Events of the 31st Brazilian Symposium on Databases - I Database Systems Industry Day Workshop (DSIDW). Brazilian Computer Society, 2016. p.106. ISBN 978-85-7669-343-7. Disponível em: $< http://www.sbbd2016.org >$.

[1] http://hadoop.apache.org

- Este artigo apresentou os primeiros resultados da prova de conceito implementada para validar a arquitetura proposta na publicação anterior. ALENCAR, A. L.; BURÉGIO, V.; FREITAS CARACIOLO, M. J.; GARCIA, V. A centralized platform on human genome for supporting clinical decisions. In: XV Congresso Brasileiro de Informática em Saúde. Sociedade Brasileira de Informática em Saúde, 2016. p.569 – 577. ISSN 2178-2857. Disponível em: $< http://www.sbis.org.br/biblioteca_virtual/cbis/Anais_C$

- Este artigo apresenta uma extensão da publicação anterior e foi publicado no *Journal of Health Informatics* por indicação do Congresso Brasileiro de Informática em Saúde. ALENCAR, A. L.; BURÉGIO, V.; FREITAS CARACIOLO, M. J.; GARCIA, V. A centralized platform on human genome for supporting clinical decisions. In: Journal of Health Informatics. Brazilian Health Informatics Society, 2016. p.569 – 577. ISSN 2175-4411. Disponível em: $< http://www.jhi-sbis.saude.ws/ojs-jhi/index.php/jhi-sbis/issue/view/72 >$.

7.5 SOFTWARES DESENVOLVIDOS

As ferramentas de software desenvolvidas neste trabalho podem ser encontradas no diretório Git disponível em: https://github.com/UFPE/genModel

REFERÊNCIAS

ABITEBOUL, S. Querying semi-structured data. In: AFRATI, F.; KOLAITIS, P. (Ed.). *Database Theory — ICDT '97*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996. p. 1–18. ISBN 978-3-540-49682-3.

ABITEBOUL, S.; BUNEMAN, P.; SUCIU, D. *Data on the Web: From Relations to Semistructured Data and XML*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000. ISBN 1-55860-622-X.

ALENCAR, A. L.; BURéGIO, V.; FREITAS CARACIOLO, M. J.; GARCIA, V. A centralized platform for access of heterogeneous data on human genome repositories for supporting clinical decisions. In: *Proceedings of the Satellite Events of the 31st Brazilian Symposium on Databases - I Database Systems Industry Day Workshop (DSIDW)*. Brazilian Computer Society, 2016, 2016. p. p.106. ISBN 978-85-7669-343-7. Disponível em: <http://www.sbbd2016.org>.

ALENCAR, A. L.; BURéGIO, V.; FREITAS CARACIOLO, M. J.; GARCIA, V. A centralized platform on human genome for supporting clinical decisions. In: *Journal of Health Informatics*. Brazilian Health Informatics Society, 2016, 2016. p. p.569 – 577. ISSN 2175-4411. Disponível em: <http://www.jhi-sbis.saude.ws/ojs-jhi/index.php/jhi-sbis/issue/view/72>.

ALENCAR, A. L.; BURéGIO, V.; FREITAS CARACIOLO, M. J.; GARCIA, V. A centralized platform on human genome for supporting clinical decisions. In: *XV Congresso Brasileiro de Informática em Saúde*. Sociedade Brasileira de Informática em Saúde, 2016, 2016. p. p.569 – 577. ISSN 2178-2857. Disponível em: <http://www.sbis.org.br/biblioteca_virtual/cbis/Anais_CBIS_2016_Artigos_Completos.pdf>.

ATKINSON, C.; KüHNE, T. Model-driven development: A metamodeling foundation. *IEEE Softw.*, IEEE Computer Society Press, Los Alamitos, CA, USA, v. 20, n. 5, p. 36–41, set. 2003. ISSN 0740-7459.

ATZENI, P.; BELLOMARINI, L.; BUGIOTTI, F.; CELLI, F.; GIANFORME, G. A runtime approach to model-generic translation of schema and data. *Information Systems*, v. 37, n. 3, p. 269 – 287, 2012. ISSN 0306-4379.

ATZENI, P.; BUGIOTTI, F.; ROSSI, L. Uniform access to nosql systems. *Information Systems*, v. 43, p. 117 – 133, 2014. ISSN 0306-4379.

ATZENI, P.; CAPPELLARI, P.; GIANFORME, G. Midst: Model independent schema and data translation. In: *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*. New York, NY, USA: ACM, 2007. (SIGMOD '07), p. 1134–1136. ISBN 978-1-59593-686-8.

BALKO, S.; LANGE, M.; SCHNEE, R.; SCHOLZ, U. *BioDataServer: an Applied Molecular Biological Data Integration Service*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004. 140–155 p. ISBN 978-3-540-24745-6.

BASILI, V.; GREEN, S. Software process evolution at the sel. *IEEE Softw.*, IEEE Computer Society Press, Los Alamitos, CA, USA, v. 11, n. 4, p. 58–66, jul. 1994. ISSN 0740-7459.

BASILI, V. R.; CALDIERA, G.; ROMBACH, D. H. *The Goal Question Metric Approach*. [S.l.]: John Wiley & Sons, 1994. I.

BATINI, C.; SCANNAPIECO, M. *Data Quality: Concepts, Methodologies and Techniques (Data-Centric Systems and Applications)*. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN 3540331727.

BERNSTEIN, P. A.; HAAS, L. M.; JARKE, M.; RAHM, E.; WIEDERHOLD, G. Panel: Is generic metadata management feasible? In: *Proceedings of the 26th International Conference on Very Large Data Bases*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000. (VLDB '00), p. 660–662. ISBN 1-55860-715-3.

BERNSTEIN, P. A.; MELNIK, S. Model management 2.0: Manipulating richer mappings. In: *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*. New York, NY, USA: ACM, 2007. (SIGMOD '07), p. 1–12. ISBN 978-1-59593-686-8. Disponível em: <http://doi.acm.org/10.1145/1247480.1247482>.

BEX, G. J.; NEVEN, F.; SCHWENTICK, T.; VANSUMMEREN, S. Inference of concise regular expressions and dtds. *ACM Trans. Database Syst.*, ACM, New York, NY, USA, v. 35, n. 2, p. 11:1–11:47, maio 2010. ISSN 0362-5915.

BEZIVIN, J.; NO, C. D. G.; NANTES, J. B.; GERBé, O.; MONTRéAL, H. *Towards a Precise Definition of the OMG/MDA Framework*. 2001.

BOOCH, G. *Object-oriented Analysis and Design with Applications (2Nd Ed.)*. Redwood City, CA, USA: Benjamin-Cummings Publishing Co., Inc., 1994. ISBN 0-8053-5340-2.

BRAMBILLA, M.; CABOT, J.; WIMMER, M. *Model-Driven Software Engineering in Practice*. San Rafael, CA: Morgan e Claypool, 2012. v. 1. (Synthesis Lectures on Software Engineering, v. 1). ISBN 978-1-60845-882-0.

BUNEMAN, P. Semistructured data. In: *Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*. New York, NY, USA: ACM, 1997. (PODS '97), p. 117–121. ISBN 0-89791-910-6.

CHHIBBER, S.; APOSTOLAKIS, G.; OKRENT, D. A taxonomy of issues related to the use of expert judgments in probabilistic safety studies. In: *Reliability engineering systems safety*. [S.l.: s.n.], 1992. p. 27–45.

CHIDAMBER, S. R.; KEMERER, C. F. A metrics suite for object oriented design. *IEEE Trans. Softw. Eng.*, IEEE Press, Piscataway, NJ, USA, v. 20, n. 6, p. 476–493, jun. 1994. ISSN 0098-5589. Disponível em: <http://dx.doi.org/10.1109/32.295895>.

CHILLÓN, A. H.; MORALES, S. F.; RUIZ, D. S.; MOLINA, J. G. Exploring the visualization of schemas for aggregate-oriented nosql databases. In: *ER Forum/Demos*. [S.l.: s.n.], 2017.

CHIN, L.; ANDERSEN, J.; FUTREAL, P. Cancer genomics: from discovery science to personalized medicine. *Nat Med. 2011;17(3)*, p. 297–303, 2011.

COLAZZO, D.; GHELLI, G.; SARTIANI, C. Typing Massive JSON Datasets. In: *International Workshop on Cross-model Language Design and Implementation (XLDI)*. Copenhagen, Denmark: [s.n.], 2012.

COOKE, R. *Experts in Uncertainty: Opinion and Subjective Probability in Science*. [S.l.]: Oxford University Press, Inc, 1991. ISBN 0195064658.

EASTERBROOK, S.; SINGER, J.; STOREY, M.-A.; DAMIAN, D. *Selecting Empirical Methods for Software Engineering Research*. [S.l.]: Springer, 2007.

FAGIN, R.; KOLAITIS, P. G.; POPA, L. Data exchange: Getting to the core. *ACM Trans. Database Syst.*, ACM, New York, NY, USA, v. 30, n. 1, p. 174–210, mar. 2005. ISSN 0362-5915.

FAVRE, J.-M.; NGUYEN, T. Towards a megamodel to model software evolution through transformations. *Electronic Notes in Theoretical Computer Science*, v. 127, n. 3, p. 59 – 74, 2005. ISSN 1571-0661. Proceedings of the Workshop on Software Evolution through Transformations: Model-based vs. Implementation-level Solutions (SETra 2004).

FRANKEL, D. *Model Driven Architecture - Applying MDA to Enterprise Computing*. New York, NY, USA: John Wiley & Sons, Inc., 2002. ISBN 0471319201.

GARSON, G. D. *Statnotes: Topics in Multivariate Analysis*. 2018.

GARZA, M.; FIOL, G. D.; TENENBAUM, J.; WALDEN, A.; ZOZUS, M. N. Evaluating common data models for use with a longitudinal community registry. *Journal of Biomedical Informatics*, v. 64, p. 333 – 341, 2016. ISSN 1532-0464. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1532046416301538>.

GOYAL, G.; SINGH, K.; RAMKUMAR, K. R. A detailed analysis of data consistency concepts in data exchange formats (json xml). In: *2017 International Conference on Computing, Communication and Automation (ICCCA)*. [S.l.: s.n.], 2017. p. 72–77.

GREENFIELD, J.; SHORT, K. Software factories - assembling applications with patterns, models, frameworks and tools. In: *Companion of the 18th Annual ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications*. New York, NY, USA: ACM, 2003. (OOPSLA '03), p. 16–27. ISBN 1-58113-751-6.

HAAS, L. M.; HERNáNDEZ, M. A.; HO, H.; POPA, L.; ROTH, M. Clio grows up: From research prototype to industrial tool. In: *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*. New York, NY, USA: ACM, 2005. (SIGMOD '05), p. 805–810. ISBN 1-59593-060-4. Disponível em: <http://doi.acm.org/10.1145/1066157.1066252>.

HEGEWALD, J.; NAUMANN, F.; HERSCHEL, M. Xstruct: Efficient schema extraction from multiple and large xml documents. *22nd International Conference on Data Engineering Workshops (ICDEW'06)*, p. 81–81, 2006.

IDC. *The Digital Universe Driving Data Grouth in Healthcare: Challenge and Opportunities for IT*. 2014.

JANGA, P.; DAVIS, K. C. Schema extraction and integration of heterogeneous xml document collections. In: CUZZOCREA, A.; MAABOUT, S. (Ed.). *Model and Data Engineering*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. p. 176–187. ISBN 978-3-642-41366-7.

KAHN, M. G.; BATSON, D.; SCHILLING, L. M. Data model considerations for clinical effectiveness researchers. *Medical care*, v. 50, n. 0, p. 10.1097/MLR.0b013e318259bff4, 07 2012.

KAPLUNOVSKY, A. S. Why using factor analysis? (dedicated to the centenary of factor analysis). 2009.

KELLY, S.; TOLVANEN, J.-P. *Domain-Specific Modeling: Enabling Full Code Generation*. [S.l.]: Wiley, 2008. ISBN 978-0-470-03666-2.

KIM, J.-O.; MUELLER, C. W. *Factor analysis: Statistical methods and practical issues*. First edition. [S.l.]: Sage Publications, 1978. ISBN 978-0803911659.

KIM, J.-O.; MUELLER, C. W. *Introduction to factor analysis - what it is and how to do it*. First edition. [S.l.]: Sage Publications, 1978. ISBN 978-0803911659.

KITCHENHAM, B. Large-scale software engineering questions â expert opinion or empirical evidence? *IET Software*, Institution of Engineering and Technology, v. 1, p. 161–171(10), October 2007. ISSN 1751-8806. Disponível em: <http://digital-library.theiet.org/content/journals/10.1049/iet-sen_20060052>.

KLEPPE, A. G.; WARMER, J.; BAST, W. *MDA Explained: The Model Driven Architecture: Practice and Promise*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2003. ISBN 032119442X.

KROGSTIE, J. *Model-Based Development and Evolution of Information Systems - A Quality Approach*. [S.l.]: Springer Publishing Company, Incorporated, 2012. ISBN 1447129350, 9781447129356.

KROGSTIE, J. Quality of conceptual data models. In: *ICISO 2013 Proceedings*. [S.l.]: SciTePress, 2013, 2013. p. p. 165–174.

KUHNE, T. Matters of (meta-) modeling. *Software Systems Modeling*, v. 5, n. 4, p. 369–385, 2006. ISSN 1619-1374.

LI, M.; SMIDTS, C. A ranking of software engineering measures based on expert opinion. *IEEE Trans. Softw. Eng.*, IEEE Press, Piscataway, NJ, USA, v. 29, n. 9, p. 811–824, set. 2003. ISSN 0098-5589. Disponível em: <http://dx.doi.org/10.1109/TSE.2003.1232286>.

LIU, Y.; LIU, X.; YANG, L. Analysis and design of heterogeneous bioinformatics database integration system based on middleware. In: *2010 2nd IEEE International Conference on Information Management and Engineering*. [S.l.: s.n.], 2010. p. 272–275.

LIU, Y.; WANG, J.; LIU, Y.; TAN, Z. Research on data integration of bioinformatics database based on web services. *First International Conference on Networked Digital Technologies*, p. 292–296, 2009.

LUCREDIO, D.; FORTES, R. P. de M.; ALMEIDA, E. S. de; MEIRA, S. L. *"Performing Domain Analysis for Model-Driven Software Reuse"*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. 200–211 p. ISBN 978-3-540-68073-4.

MARZ, N. *Big data : principles and best practices of scalable realtime data systems*. [S.l.]: O'Reilly Media, 2013. ISBN 1617290343 9781617290343.

MASSEROLI, M.; CANAKOGLU, A.; CERI, S. Integration and querying of genomic and proteomic semantic annotations for biomedical knowledge extraction. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, v. 13, n. 2, p. 209–219, March 2016. ISSN 1545-5963.

MELLOR, S. J.; BALCER, M. *Executable UML: A Foundation for Model-Driven Architectures*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2002. ISBN 0201748045.

MELNIK, S. *Generic Model Management: Concepts and Algorithms*. [S.l.]: Springer-Verlag, 2004.

MILLER, R. J.; HAAS, L. M.; HERNáNDEZ, M. A. Schema mapping as query discovery. In: *Proceedings of the 26th International Conference on Very Large Data Bases*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000. (VLDB '00), p. 77–88. ISBN 1-55860-715-3. Disponível em: <http://dl.acm.org/citation.cfm?id=645926.671677>.

MIN, J.-K.; AHN, J.-Y.; CHUNG, C.-W. Efficient extraction of schemas for xml documents. *Information Processing Letters*, v. 85, n. 1, p. 7–12, 2003.

MOH, C.-H.; LIM, E.-P.; NG, W. K. Dtd-miner: A tool for mining dtd from xml documents. In: *WECWIS*. [S.l.: s.n.], 2000.

MOODY, D. L. The "physics"of notations: a scientific approach to designing visual notations in software engineering. In: *2010 ACM/IEEE 32nd International Conference on Software Engineering*. [S.l.: s.n.], 2010. v. 2, p. 485–486. ISSN 0270-5257.

MOODY, D. L.; SHANKS, G. G. Improving the quality of data models: empirical validation of a quality management framework. *Information Systems*, v. 28, n. 6, p. 619 – 650, 2003. ISSN 0306-4379.

NESTOROV, S.; ABITEBOUL, S.; MOTWANI, R. Extracting schema from semistructured data. *SIGMOD Rec.*, ACM, New York, NY, USA, v. 27, n. 2, p. 295–306, jun. 1998. ISSN 0163-5808.

NUREG-1150. Reactor risk reference document. 1989.

OIKAWA, M. K.; BROINIZI, M. E. B.; DERMARGOS, A.; ARMELIN, H. A.; FERREIRA, J. E. Genflow : Generic flow for integration , management and analysis of molecular biology data. In: *Scielo Genetics and Molecular Biology*. [S.l.: s.n.], 2004.

PABINGER, S.; DANDER, A.; FISCHER, M.; SNAJDER, R.; SPERK, M.; EFREMOVA, M.; KRABICHLER, B.; SPEICHER, M. R.; ZSCHOCKE, J.; TRAJANOSKI, Z. A survey of tools for variant analysis of next-generation genome sequencing data. *Briefings in Bioinformatics*, v. 15, n. 2, p. 256, 2014.

PALLANT, J. *SPSS Survival Manual*. 5th. ed. [S.l.]: Open University Press, 2013. ISBN 978-0335262588.

RUNESON, P.; HöST, M. Guidelines for conducting and reporting case study research in software engineering. *Empirical Softw. Engg.*, Kluwer Academic Publishers, Hingham, MA, USA, v. 14, n. 2, p. 131–164, abr. 2009. ISSN 1382-3256.

SADALAGE, P. J.; FOWLER, M. *NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence*. 1st. ed. [S.l.]: Addison-Wesley Professional, 2012. ISBN 0321826620, 9780321826626.

SAGIROGLU, S.; SINANC, D. Big data: A review. In: . [S.l.: s.n.], 2013. p. 42–47. Cited By (since 1996)0.

SCHNAARS, S. P. Long-range forecasting: From crystal ball to computer: J. scott armstrong, 2nd ed. (wiley, new york, 1985) [uk pound]22.95 (paper), pp. 689. *International Journal of Forecasting*, v. 2, n. 3, p. 387–390, 1986. Disponível em: <https://EconPapers.repec.org/RePEc:eee:intfor:v:2:y:1986:i:3:p:387-390>.

SEIDEWITZ, E. What models mean. *IEEE Softw.*, IEEE Computer Society Press, Los Alamitos, CA, USA, v. 20, n. 5, p. 26–32, set. 2003. ISSN 0740-7459.

SELIC, B. The pragmatics of model-driven development. *IEEE Softw.*, IEEE Computer Society Press, Los Alamitos, CA, USA, v. 20, n. 5, p. 19–25, set. 2003. ISSN 0740-7459.

SELIC, B. Personal reflections on automation, programming culture, and model-based software engineering. *Automated Software Engineering*, v. 15, n. 3, p. 379–391, 2008. ISSN 1573-7535.

SEVILLA, D.; FELICIANO, S.; GARCíA-MOLINA, J. An mde approach to generate schemas for object-document mappers. In: INSTICC. *Proceedings of the 5th International Conference on Model-Driven Engineering and Software Development - Volume 1: MODELSWARD,*. [S.l.]: SciTePress, 2017. p. 220–228. ISBN 978-989-758-210-3.

SIEGEL, J. Object management group model driven architecture (mda) mda guide rev. 2.0. 2014.

SILVA, A. R. da. Model-driven engineering: A survey supported by the unified conceptual model. *Computer Languages, Systems Structures*, v. 43, p. 139 – 155, 2015. ISSN 1477-8424.

STAHL, T.; VOELTER, M.; CZARNECKI, K. *Model-Driven Software Development: Technology, Engineering, Management*. [S.l.]: John Wiley & Sons, 2006. ISBN 0470025700.

TVERSKY, A.; KAHNEMAN, D. Judgment under uncertainty: Heuristics and biases. *Science*, v. 185, p. 1124–31, 1974.

UTTING, M.; LEGEARD, B. *Practical Model-Based Testing: A Tools Approach*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007. ISBN 0123725011, 9780080466484.

VELEGRAKIS, Y.; MILLER, R. J.; POPA, L. Mapping adaptation under evolving schemas. In: *Proceedings of the 29th International Conference on Very Large Data Bases - Volume 29*. [S.l.]: VLDB Endowment, 2003. (VLDB '03), p. 584–595. ISBN 0-12-722442-4.

VOELTER, M.; BENZ, S.; DIETRICH, C.; ENGELMANN, B.; HELANDER, M.; KATS, L. C. L.; VISSER, E.; WACHSMUTH, G. *DSL Engineering - Designing, Implementing and Using Domain-Specific Languages*. [S.l.]: dslbook.org, 2013. 1–558 p.

WANG, K.; LIU, H. Schema discovery for semistructured data. In: *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*. [S.l.]: AAAI Press, 1997. (KDD'97), p. 271–274.

WANG, L.; ZHANG, S.; SHI, J.; JIAO, L.; HASSANZADEH, O.; ZOU, J.; WANGZ, C. Schema management for document stores. *Proc. VLDB Endow.*, VLDB Endowment, v. 8, n. 9, p. 922–933, maio 2015. ISSN 2150-8097.

WEAVER, P. *Practical SSADM*. [S.l.]: FT Prentice Hall, 1993.

WHITE, T. *Hadoop - The Definitive Guide*. [S.l.]: O'Reilly, 2010. ISBN 1449389732.

WOHLIN, C.; RUNESON, P.; HöST, M.; OHLSSON, M. C.; REGNELL, B.; WESSLÉN, A. *Experimentation in Software Engineering: An Introduction*. Norwell, MA, USA: Kluwer Academic Publishers, 2000. ISBN 0-7923-8682-5.

XU, L.; EMBLEY, D. W. Combining the best of global-as-view and local-as-view for data integration. In: *ISTA*. [S.l.: s.n.], 2004.

YANG, Y.; MENEGHETTI, N.; FEHLING, R.; LIU, Z. H.; KENNEDY, O. Lenses: An on-demand approach to etl. *Proc. VLDB Endow.*, VLDB Endowment, v. 8, n. 12, p. 1578–1589, ago. 2015. ISSN 2150-8097.

YIN, R. K. *Case Study Research: Design and Methods (Applied Social Research Methods)*. Fourth edition. [S.l.]: Sage Publications, 2008. ISBN 1412960991.

YOURDON, E. *Modern structured analysis*. Englewood Cliffs, NJ [u.a.]: Prentice-Hall Internat., 1989. (Prentice-Hall international series). ISBN 013598632X.

APÊNDICE A – QUESTIONÁRIO DE AVALIAÇÃO

Avaliação GenDB Metamodel

Esta atividade tem o objetivo de avaliar o metamodelo desenvolvido com técnicas de análise de domínio e MDE para representação de bases de dados de variações de genoma.

* Required

1. **Email address** *

Qualidade Empírica e Pragmática Social

- Empírica: Lida com a compreensão e frequências de erros previsíveis quando um meta-modelo é lido ou escrito por diferentes atores (pessoas ou ferramentas) .
- Pragmática Social : correspondência entre o metamodelo e a interpretação do leitor(pessoa).
- Pragmática técnica: refere a interpretação do leitor (ferramentas) e será automaticamente avaliada.

Compreensibilidade

Define com qual facilidade os elementos e estruturas do meta-modelo podem ser compreendidos.

2. **Como você classificaria o nível de compreensão do metamodelo? Indique na escala abaixo** *

Mark only one oval.

	1	2	3	4	5	
Totalmente Incompreensível	◯	◯	◯	◯	◯	Totalmente Compreensível

Simplicidade

Verifica se o meta-modelo contém os elementos e relacionamentos mínimos possíveis ainda representando o domínio.

3. **Como você classificaria a simplicidade do metamodelo? Indique na escala abaixo** *

Mark only one oval.

	1	2	3	4	5	
Muito Complexo	◯	◯	◯	◯	◯	Muito Simples

4. **Você tem outras considerações sobre a compreensão e simplicidade do modelo? Insira aqui**

Qualidade Semântica

Define se o meta-modelo reflete o conhecimento explícito (das partes interessadas) relevante para o domínio.
É uma correspondência entre o meta-modelo e o conjunto de todas as declarações que podem ser estabelecidas sobre a situação.
Em resumo, verifica se todos os elementos do domínio são acomodados pelo meta-modelo.

Completude

Define se o metamodelo contém todas as informações e elementos necessárias para suportar os requisitos do domínio e das funcionalidades do sistema.

5. **Quais elementos (metaclasses e/ou atributos) estão faltando no metamodelo? Por que você considera isso?**

6. **Quais dos elementos do metamodelo (metaclasses e/ou atributos) são supérfluos/desnecessários? Por que você considera isso?**

7. **Como você classificaria a completude do metamodelo? Indique na escala abaixo** *
   Mark only one oval.

	1	2	3	4	5	
Totalmente Incompleto	◯	◯	◯	◯	◯	Totalmente Completo

Validade

Define se o meta-modelo apresenta de maneira correta (válida) todas as regras de negócios que se aplicam ao domínio.

8. **Quais elementos do metamodelo (metaclasses e/ou atributos) podem ter sido definidos de forma incorreta? Por que você considera isso?**

9. **Como você classificaria a validade do metamodelo? Indique na escala abaixo** *
   Mark only one oval.

	1	2	3	4	5	
Totalmente Inválido	◯	◯	◯	◯	◯	Totalmente Válido

Integridade

Define se as associações no metamodelo correspondem às necessidades específicas do projeto.

10. **Quais restrições/associações no metamodelo não estão de acordo com as regras de negócio do domínio? Por que você considera isso?**

11. **Como você classificaria a integridade do metamodelo? Indique na escala abaixo** *
    Mark only one oval.

	1	2	3	4	5	
Nada Íntegro	◯	◯	◯	◯	◯	Totalmente Íntegro

Integração

Define a consistência do metamodelo com as necessidades corporativas ou sistemas existentes.

12. **Quais conflitos existem entre o metamodelo e os sistemas existentes? Por que você considera isso?**

13. **Quais elementos do metamodelo (metaclasses e/ou atributos) possuem duplicidade em relação aos sistemas existentes? Por que você considera isso?**

14. **Como você classificaria a capacidade de integração do metamodelo? Indique na escala abaixo** *

Mark only one oval.

	1	2	3	4	5	
Nada Integrável	◯	◯	◯	◯	◯	Totalmente Integrável

Reuso

Define o nível de reuso do metamodelo

15. **Este metamodelo pode ser reutilizado? Se sim, indique e descreva as possibilidades de reuso e aplicação**

16. **Como você classificaria o grau de reuso do metamodelo? Indique na escala abaixo** *

Mark only one oval.

	1	2	3	4	5	
Nada Reutilizável	◯	◯	◯	◯	◯	Totalmente reutilizável

Minimalização

Define se nenhum requisito é representado mais de uma vez

17. **Quais metaclasses do metamodelo estão redundantes? Por que você considera isso?**

18. **Quais atributos do metamodelo estão redundantes? Por que você considera isso?**

19. **Como você classificaria o minimalismo do metamodelo? Indique na escala abaixo** *
Mark only one oval.

	1	2	3	4	5	
Insatisfatório	◯	◯	◯	◯	◯	Totalmente Satisfatório

Qualidade Organizacional

Define se todas as declarações do metamodelo contribuem para o cumprimento dos objetivos da modelagem, e se todos os objetivos de modelagem são abordados através do metamodelo.

Flexibilidade

Definido em termos de extensibilidade e adaptabilidade.
Verifica o nível de facilidade em que o metamodelo pode se adaptar às mudanças sem alterar a sua integridade.

20. **Quais elementos do metamodelo (metaclasses e/ou atributos) estão sujeitos a mudanças? Por que você considera isso?**

21. **Qual o custo (tempo de treinamento e programação) das mudanças apontadas na questão anterior? Faça uma estimativa**

22. **Como você classificaria a flexibilidade do metamodelo? Indique na escala abaixo** *
Mark only one oval.

	1	2	3	4	5	
Nada Flexível	◯	◯	◯	◯	◯	Totalmente Flexível

Implementabilidade

Definido em termos de: (1) Aplicabilidade - diversidade de usos do metamodelo. (2) Custo - a facilidade com que o metamodelo pode ser implementado dentro dos recursos do projeto.

23. **Quais outras possibilidades de uso do metamodelo você conhece? Para quais outras situações (sistemas, modelos de DB, ...) ele pode ser aplicado? Por que você considera isso?**

24. **Qual o tempo de programação necessário para aplicar o metamodelo ao ambiente do projeto? Faça uma estimativa**

25. **Qual o tempo de treinamento necessário para aplicar o metamodelo ao ambiente do projeto? Faça uma estimativa.**

26. **Como você classificaria a implementabilidade (aplicação do metamodelo em outros usos)? Indique na escala abaixo.** *

Mark only one oval.

1	2	3	4	5
◯	◯	◯	◯	◯

☐ Send me a copy of my responses.

APÊNDICE B – EXERCÍCIO PARA OBSERVAÇÃO

Observação

Para este exercício considere o exemplo de esquema da entidade ilustrado como uma árvore na Figura 1. A figura apresenta os Atributos como nós(círculos) e tipos de dados como elementos folhas(retângulos). Os nós que possuem atributos embutidos possuem outros nós filhos.

Atividade – Usando a Ferrameta de modelagem, crie o modelo referente ao esquema da Figura 1 até o nível 3 de profundidade da árvore. Adote *ClinVar* para o nome da fonte (*Source*) e *ClinVarFullRelease_2018-05* para o nome da versão (*SourceVersion*). Após a elaboração do modelo execute a rotina de validação disponível na ferramenta para verificar elementos que possam estar incorretos ou incompletos.

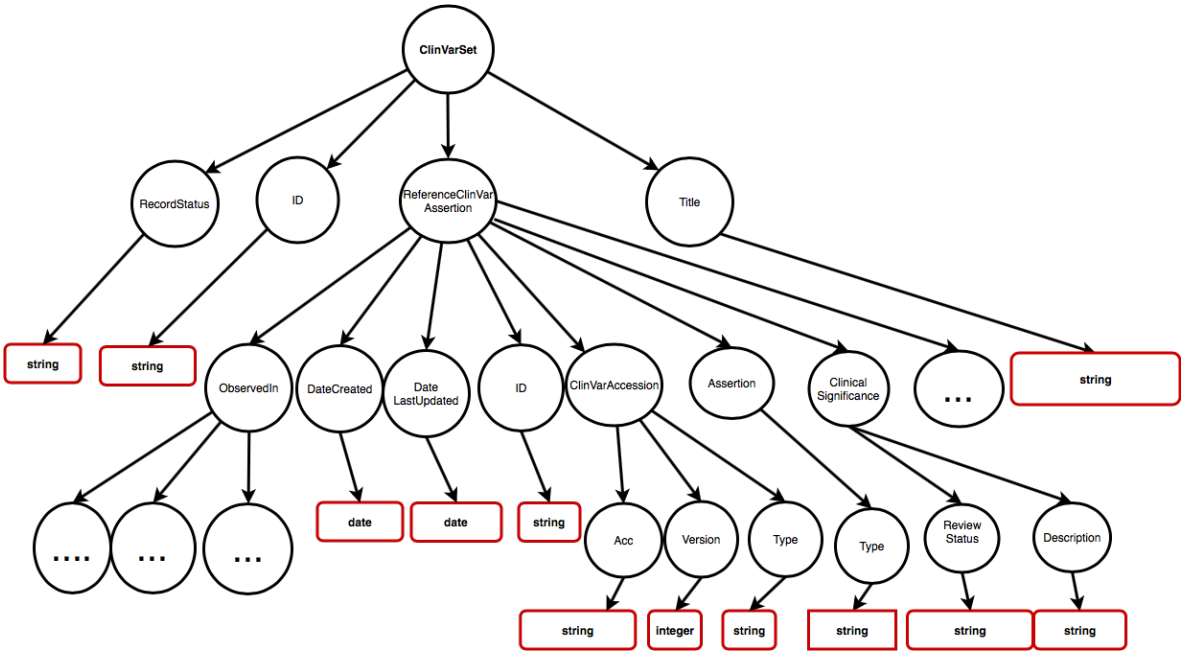

Figura 1 - Exemplo de esquema

APÊNDICE C – FORMULÁRIO DE OBSERVAÇÃO

Formulário de observação

Data : _____/_____/_____ Hora: _____:_____

Tempo de treinamento: _____:_____

Autor: Andreza Leite

Participante Observado: _____

1. Descrição dos códigos observados.
   - TM –Tempo de Modelagem: quantidade de minutos gastos para executar a tarefa de modelagem.
   - EC – Erros de compreensão: quantidade de elementos mal compreendidos/interpretados pelos usuários.
   - CO – Consultas ao observador: quantidade de vezes que o observador foi consultado para solução de dúvidas.
   - DNV – Diagramas Não Validados: diagramas não aprovados pela validação da ferramenta.
   - EA – Erros de aplicação: quantidade de elementos aplicados de maneira incorreta no modelo. Erros identificados pela validação da ferramenta.
   - DI – Diagramas incompletos: quantidade de elementos com requisitos (atributos ou sub-elementos) obrigatórios não atendidos pelo modelo.
   - EF – Erros da ferramenta: quantidade de erros permitidos pela ferramenta. Erros não identificados pela validação da ferramenta.
   - PU – Problemas de Usabilidade: problemas de uso e manuseio dos elementos da interface gráfica.

Tempo	**Código**	**Notes**

APÊNDICE D – TREINAMENTO DOS PARTICIPANTES

Avaliação do meta-modelo para representação de bases de dados de variações de genoma.

Esta atividade tem o objetivo de avaliar o meta-modelo desenvolvido com técnicas de análise de domínio e MDE. Os paradigmas existentes em MDE compartilham os mesmos princípios básicos:

(1) modelos são usados para representar aspectos de um sistema de software em algum nível de abstração;
(2) os modelos são exemplos(instâncias) de meta-modelos (ou estão em conformidade com eles);
(3) as transformações do modelo fornecem automação no processo de desenvolvimento de software; e
(4) modelos podem ser expressos por meio de linguagens de modelagem.

Além disso, há quatro cenários principais em que as técnicas de MDE podem ser aplicadas:

(1) construção de novas aplicações de software,
(2) modernização de software (por exemplo, reengenharia de software),
(3) uso de modelos em tempo de execução para representar o contexto ou executar o sistema, e
(4) integração de ferramentas de software.

Nesta pesquisa, desenvolvemos soluções baseadas em modelos para lidar com problemas relacionados aos dois primeiros cenários: (1)engenharia reversa para descoberta de esquemas de bases de dados em forma de modelos e (2) desenvolvimento de banco de dados a partir dos modelos extraídos.

Vamos apresentar as definições para alguns desses conceitos essenciais de MDE:

Sistema – um conceito abstrato para designar uma aplicação de software, uma plataforma de software ou outro artefato de software.

Modelo – uma abstração de um sistema em estudo que pode já existir ou se destina a existir no futuro.

Metamodelo – um modelo que define a estrutura de uma linguagem de modelagem.

1. O Metamodelo

Conhecendo estes conceitos. Analise o meta-modelo definido para representação das bases de dados de variações de genoma ilustrado na figura 1.

A representação completa (metaclasse *GenDB*) pode possuir várias fontes de dados (metaclasse *Sources*) que por sua vez podem possuir várias versões (metaclasse *SourceVersion*). Sobre cada versão são capturados o nome, o id (*versionId*), a data de criação (*createDate*), e o status que indica se esta versão é um lançamento histórico (*release*) ou a última versão lançada (*latest*). Cada versão pode possuir várias entidades (metaclasse *Entity*). Como mencionado na sessão anterior uma entidade representa as afirmações (que pode ser comparada a uma entidade no modelo ER). Sobre as afirmações capturamos o nome(*name*) e as datas de criação *createDate*) e de atualização (*updateDate*) da afirmação. Todas essas informações permitem a rastreabilidade e versionamento dos dados.

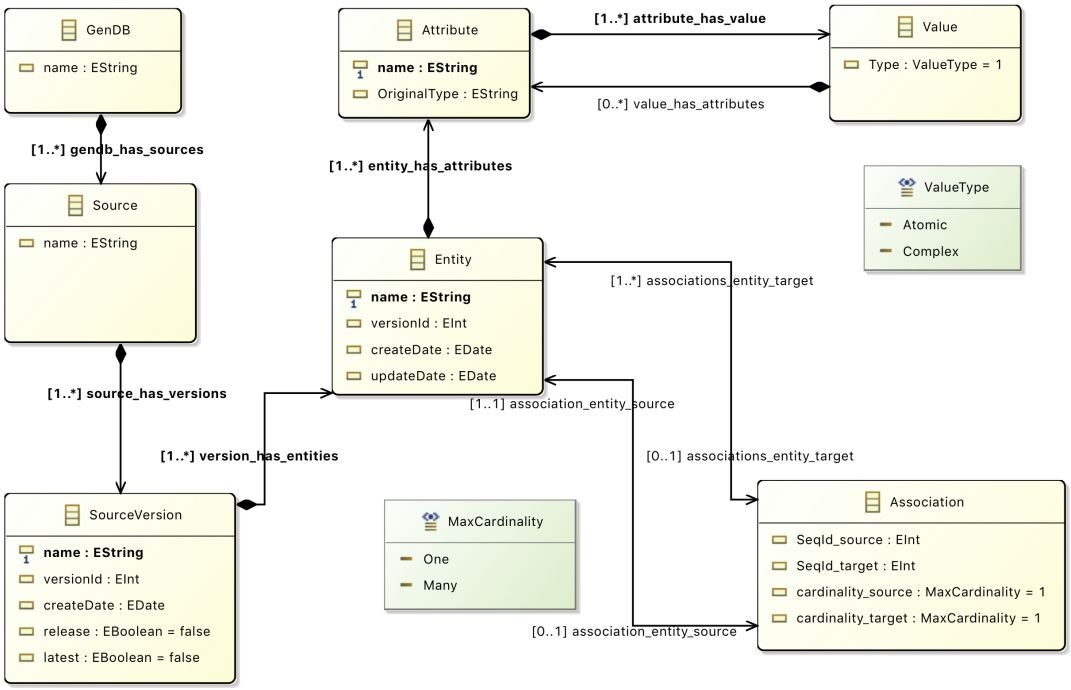

Figura 1: O metamodelo

A metaclasse *Attribute* representa as propriedades de cada afirmação (que pode ser comparada a um atributo no modelo ER). Este atributo pode ser um campo de dados simples ou uma associação. Cada atributo possui, além do par (nome, valor), o tipo de dado original do dado capturado pelo atributo *OriginalType*. O valor de um atributo (metaclasse *Value*) pode ser um valor atômico ou complexo (i. e., uma lista de valores ou outros atributos aninhados). O tipo do valor é definido pelo tipo de dado ENUM *ValueType* e capturado pelo atributo *Type*. Para um valor atômico o tipo do valor é indicado com *Type:ValueType=Atomic*, por exemplo. Para um valor complexo o tipo do valor é indicado como *Type:ValueType=Complex*, por exemplo, e a relação *value_has_attributes* indica os atributos que estão aninhados. O valor do tipo complexo permite a representação de sub-atributos e objetos agregados (que podem ser comparados a documentos embutidos em bancos de dados orientados a documento).

Uma associação (metaclasse *Association*) é utilizada para representar referências entre entidades. Esta pode ser comparada a uma chave estrangeira no modelo relacional ou uma referência a um documento no modelo orientado a documentos.

Os relacionamentos *associations_entity_target* e *associations_entity_source* são necessários para tornar a referência bidirecional e, além de especificar, permitir a navegação partindo tanto da origem(*source*) quanto do destino (*target*). A cardinalidade desta associacão é definida pelo tipo de dado ENUN *MaxCadinality* e capturada pelos atributos *cardinality_source* e *cardinality_target*.

É importante destacar que os construtores presentes neste metamodelo foram definidos com o intuito de representar esquemas de bases de dados de diversos modelos mas aplicamos esta solução inicialmente para inferência de esquemas das bases XML e criação de bases de dados orientadas a Documento (MongoDB).

2. A ferramenta de modelagem

Nesta pesquisa também foi projetada uma arquitetura de domínio e uma implementação de referência foi elaborada para validação e aplicação do metamodelo proposto. A Figura 2 apresenta o fluxo de dados na implementação de referência da arquitetura proposta.

Como podemos ver, foi desenvolvida uma ferramenta case para visualização e edição dos esquemas(modelos) inferidos no processo de descoberta de esquema.

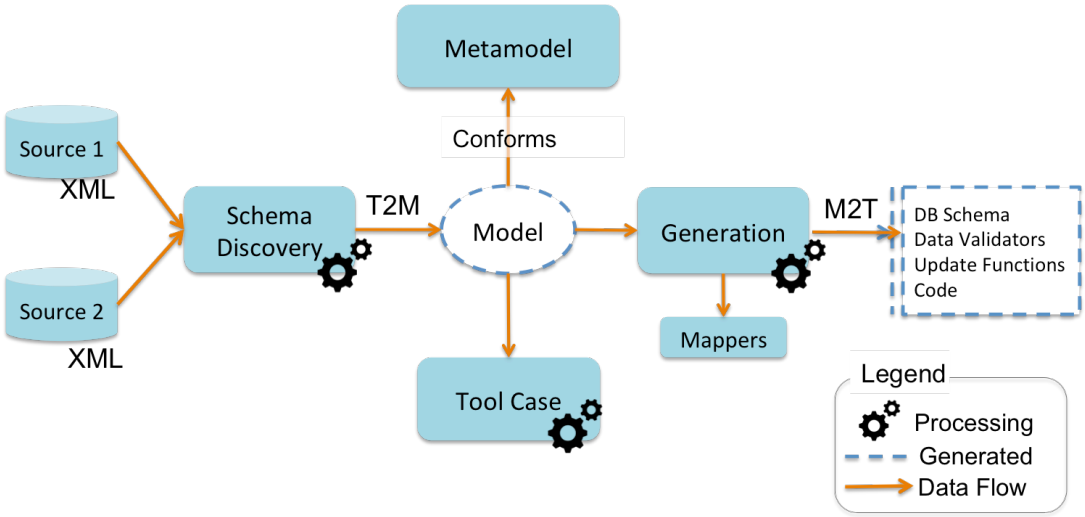

Figura 2 - Fluxo de dados na implementação de referencia da solução

A notação definida para a modelagem é baseada em símbolos gráficos, com o intuito de se ter uma representação visual para cada elemento dos diagramas, e duas técnicas visuais para demonstrar a hierarquia entre os elementos: (1) uma organização baseada em contêineres, e outra (2) que organiza os elementos do modelo como uma árvore. As Figuras 3 e 4 apresentam exemplos dos diagramas correspondentes a estas duas técnicas .

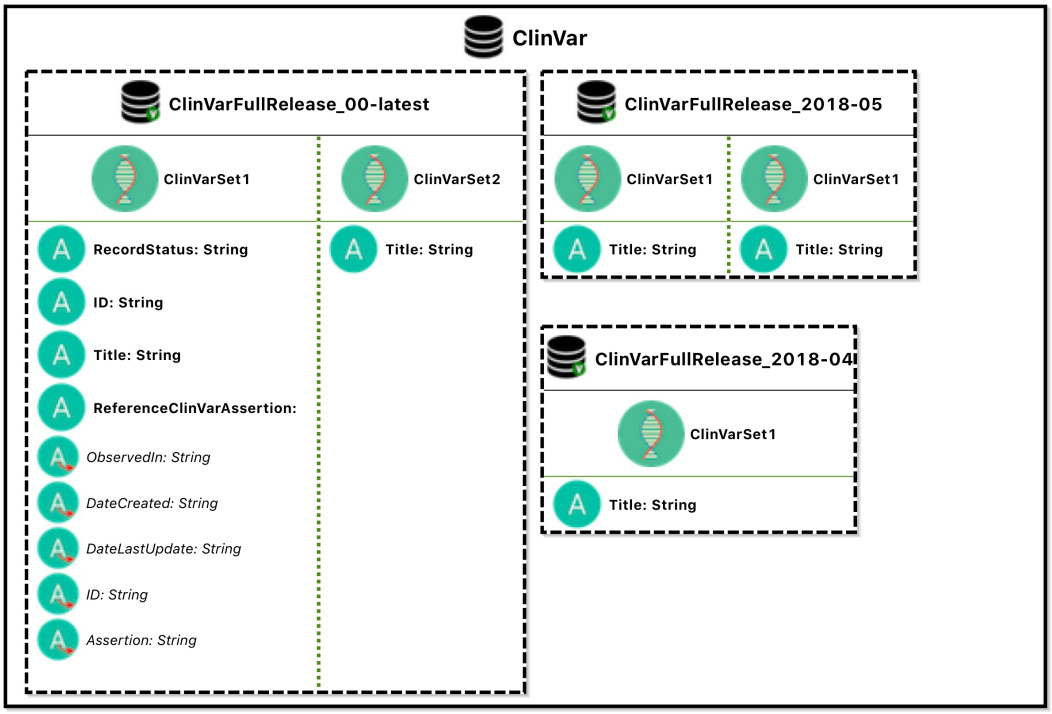

Figura 3 – Diagrama baseado em contêineres para a fonte ClinVar

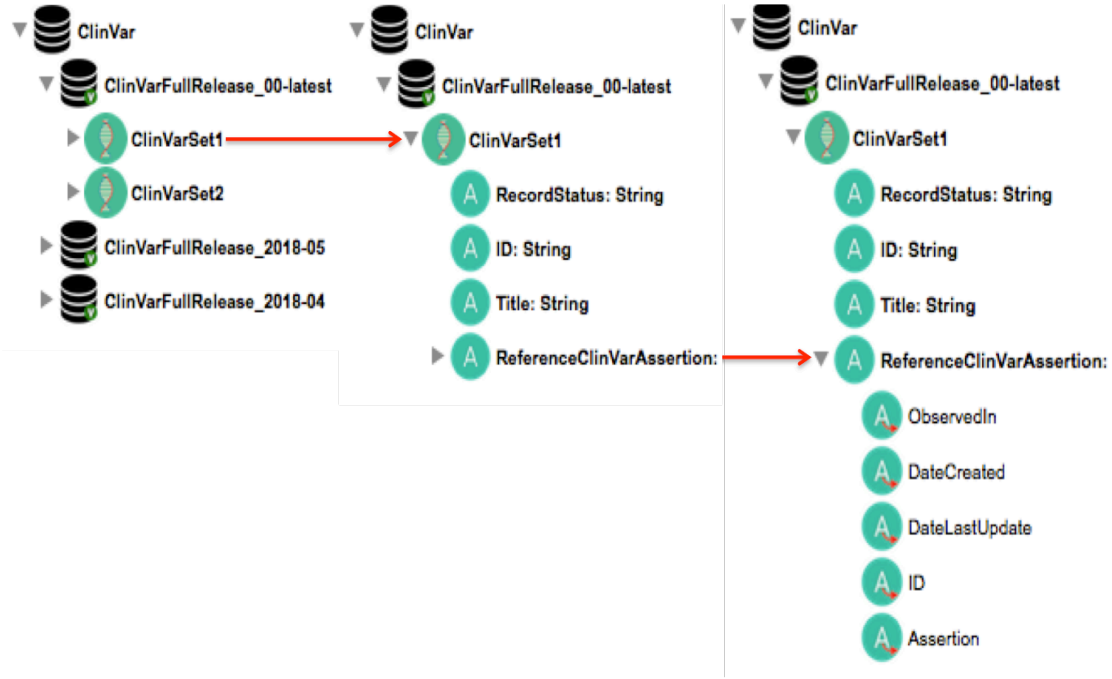

Figura 4 – Diagrama em formato de árvore pra a fonte ClinVar.

3. Avaliação

3.1. Questionário

A avaliação dos metamodelo seguira o método qualitativo baseado em survey com especialistas do domínio. Para este responda ao questionários disponível em no link: **https://bit.ly/2Lccft6**

.

Observação

Para este exercício considere o exemplo de esquema da entidade ilustrado como uma árvore na Figura 1. A figura apresenta os Atributos como nós(círculos) e tipos de dados como elementos folhas(retângulos). Os nós que possuem atributos embutidos possuem outros nós filhos.

Atividade – Usando a Ferrameta de modelagem, crie o modelo referente ao esquema da Figura 1 até o nível 3 de profundidade da árvore. Adote *ClinVar* para o nome da fonte (*Source*) e *ClinVarFullRelease_2018-05* para o nome da versão (*SourceVersion*). Após a elaboração do modelo execute a rotina de validação disponível na ferramenta para verificar elementos que possam estar incorretos ou incompletos.

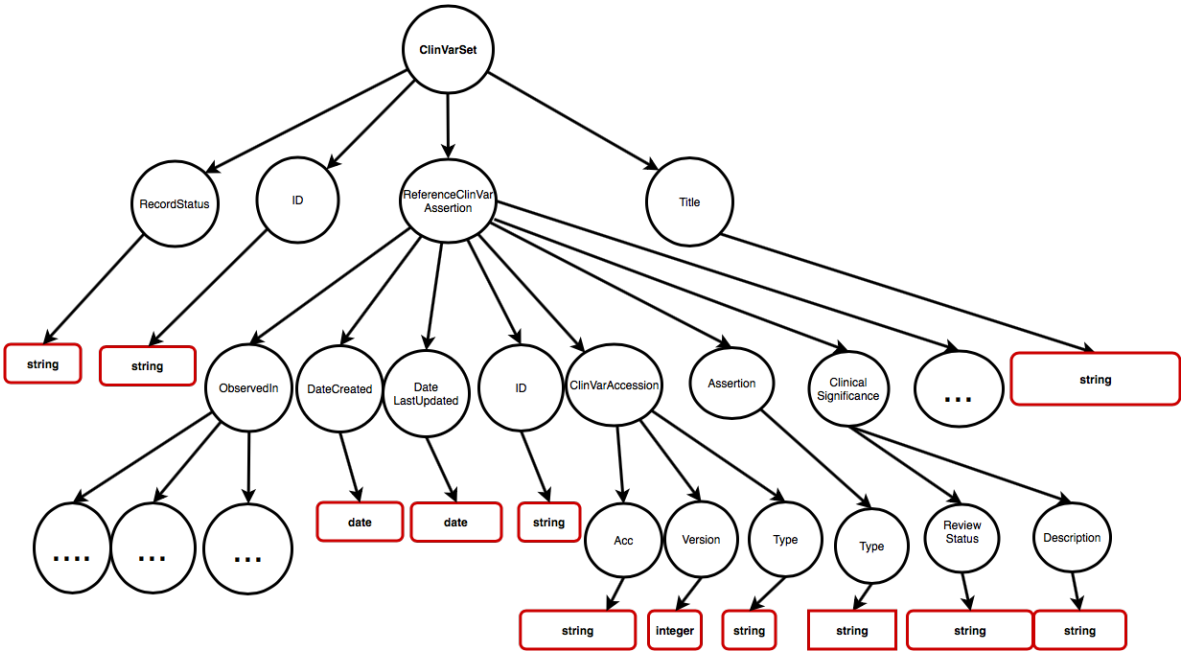

Figura 1 - Exemplo de esquema