Representação Ontológica de Frameworks de Mapeamento Objeto/Relacional

Félix Luiz Zanetti, Camila Zacche de Aguiar, Vítor E. Silva Souza

[1]Núcleo de Estudos em Modelagem Conceitual e Ontologias (NEMO)
Universidade Federal do Espirito Santo, Brasil
Av. Fernando Ferrari, 514 - Goiabeiras - Vitória, ES - 29075-910

{feluzan, camila.zacche.aguiar}@gmail.com, vitorsouza@inf.ufes.br

Resumo. *A utilização de frameworks de mapeamento objeto/relacional (object/relational mapping, ou ORM) é estado-da-prática no desenvolvimento de software, com diversos frameworks ORM para diferentes linguagens de programação orientadas a objetos. Contudo, até onde sabemos, não há uma definição formal dos conceitos abordados por tais frameworks, que poderia servir de base para tarefas de interoperabilidade semântica (ex.: migração de código) ou na definição de smells arquiteturais independentes de framework. Neste trabalho, apresentamos a ORM-O, uma ontologia de referencia de domínio que visa representar a semântica do mapeamento objeto/relacional quando frameworks ORM são utilizados no desenvolvimento de software.*

Abstract. *The use of object/relational mapping (ORM) frameworks is the state-of-practice in software development, with several ORM frameworks for different object-oriented programming languages. However, to our knowledge, there is no formal definition of the concepts related to these frameworks, which could serve as basis for semantic interoperability tasks (e.g., code migration) or in the definition of architectural smells independently of the chosen framework. In this work, we present ORM-O, a domain reference ontology that aims to identify and represent the semantics of object/relational mapping when such frameworks are used in software development.*

1. Introdução

O surgimento dos paradigmas Orientado a Objetos e Relacional e sua adoção conjunta no desenvolvimento de software deu destaque ao problema da Impedância Objeto/Relacional [Bauer and King 2004], ocasionado pela diferença de abordagem entre eles. Hoje, a utilização de *frameworks* de mapeamento entre as duas abordagens é estado-da-prática devido à eficiência e segurança que proporcionam ao desenvolvimento [Teixeira 2017].

Apesar de sua popularidade, até onde sabemos, não há uma definição formal dos conceitos abordados por esses *frameworks*. Portanto, apresentamos neste artigo a Ontologia de *Frameworks* de Mapeamento Objeto/Relacional (*Object/Relational Mapping Ontology* – ORM-O), uma ontologia de referência no domínio de *frameworks* ORM que visa identificar e representar a semântica do mapeamento objeto/relacional. A ontologia foi desenvolvida seguindo o método SABiO [Falbo 2014] e modelada usando OntoUML [Guizzardi 2005]. Sua avaliação foi realizada por meio de atividades de verificação e validação,

respondendo às questões de competência levantadas previamente e instanciando conceitos da ontologia utilizando trechos de código de mapeamento objeto/relacional, usando um *framework* ORM bastante popular.

ORM-O está sendo construída no escopo de um projeto que visa criar uma rede de ontologias sobre *frameworks* de desenvolvimento de software.[1] Tais ontologias nos permitirão automatizar tarefas de interoperabilidade semântica, como migração de código entre *frameworks*, ou ainda especificar *smells* na arquitetura do software de forma independente de *framework* ou linguagem, por exemplo. Como prova de conceito, desenvolvemos uma ferramenta de migração que converte código de um *framework* ORM para outro (em plataformas diferentes), utilizando a ontologia como *interlingua*.

As demais seções deste trabalho estão organizadas da seguinte maneira: a Seção 2 resume as questões geradas pela impedância objeto/relacional e sua solução pelo uso de *frameworks*; a Seção 3 introduz as ontologias de referência utilizadas como base na construção da ORM-O; a Seção 4 apresenta a ORM-O; a Seção 5 descreve a avaliação da ontologia por meio de verificação e validação; a Seção 6 compara a ORM-O com trabalhos relacionados; e, por fim, a Seção 7 traz as considerações finais.

2. Frameworks de Mapeamento Objeto/Relacional

No contexto do desenvolvimento de software, dentre os paradigmas já propostos, o Orientado a Objetos (OO) e o Relacional sustentam-se até hoje pela eficiência que cada um apresenta [Teixeira 2017]. Enquanto o paradigma Relacional provou ser popular para o desenvolvimento de bancos de dados, o paradigma OO tem sustentado várias linguagens de programação e métodos de desenvolvimento de software [Ireland et al. 2009].

No paradigma Relacional as informações são armazenadas na forma de tabelas, constituídas por linhas (tuplas) e colunas (atributos) [Date 2004]. Além disso, tabelas se relacionam entre si por meio de chaves. Por outro lado, no paradigma OO as informações são armazenadas em memória na forma de objetos contendo características e comportamentos [Teixeira 2017], definidos por meio de classes e relacionados por referências.

Comumente, esses paradigmas são utilizados em conjunto: uma linguagem OO é utilizada na codificação do programa enquanto o armazenamento das informações é feito em sistemas de banco de dados relacional. Assim, objetos são convertidos em tuplas para serem armazenados em tabelas e consultas são realizadas no banco de dados para que os objetos possam ser recuperados. No entanto, o uso combinado e as diferenças de abstração, foco e linguagem levam a uma série de problemas [Ireland et al. 2009], conhecidos como Impedância Objeto/Relacional [Bauer and King 2004].

Para auxiliar o desenvolvimento e minimizar problemas, o mapeamento objeto/-relacional é delegado a um *framework*, i.e., conjunto de códigos que fornece solução para algum problema específico. O uso de *frameworks* se tornou estado-da-prática, pois reduz consideravelmente o tempo de desenvolvimento de um projeto por reutilizar código já desenvolvido, testado e documentado por terceiros [Souza et al. 2009].

O objetivo é que, ao invés de fazer manualmente o mapeamento objeto/relacional, o desenvolvedor utilize um *framework* ORM (*Object/Relational Mapping*) que, a partir

[1] `https://nemo.inf.ufes.br/projects/sfwon/`

de determinadas configurações, se encarrega do mapeamento dos objetos para tabelas e colunas ou a recuperação dos objetos através de consultas. Existem diferentes *frameworks* ORM para diversas linguagens de programação OO como, por exemplo: Hibernate (Java), EclipseLink (Java), Django (Python), SQLAlchemy (Python), ODB (C++), QxOrm (C++), dentre outros.

Para o uso destes *frameworks*, é necessária a inclusão de bibliotecas, pacotes e arquivos de configuração, além de adaptações ao código desenvolvido. Os *frameworks* de Java obedecem ao padrão JPA (*Java Persistence API*) e utilizam trechos de código precedidos de `@` como anotações para indicação do comportamento de classes e atributos em relação aos *frameworks*. Em Python os *frameworks* possuem classes a serem estendidas ou instanciadas e métodos a serem utilizados. Em C++ são utilizados a diretiva `pragma` e os *templates* das bibliotecas dos *frameworks*.

3. Fundamentação Ontológica

A ORM-O foi construída com base em duas ontologias preexistentes: a Ontologia de Sistema de Banco de dados Relacionais (*Relational Database System Ontology* – RDBS-O) [de Aguiar et al. 2018], que representa a estrutura do banco de dados relacional, e a Ontologia de Código Orientado a Objetos (*Object-Oriented Code Ontology* – OOC-O) [de Aguiar et al. 2019], que representa os conceitos fundamentais presentes no código-fonte orientado a objetos.

A Figura 1 apresenta o fragmento da ontologia OOC-O reutilizado em ORM-O. Um **Object-Oriented Source Code** é constituído de **Physical Modules**, que por sua vez é composto por **Classes**. Uma **Class** pode ser uma **Extendable Class** quando está disponível para ser estendida por outras classes, tal como uma classe que assume o papel de subclasse ao definir uma relação de herança com uma superclasse. Uma **Class** pode ser composta de **Members**, tal como **Member Variable**, variável que pertence a uma classe. Em particular, quando representa o estado particular de um objeto, trata-se de uma **Instance Variable**. Uma **Variable** pode ser caracterizada por um **Value Type**.

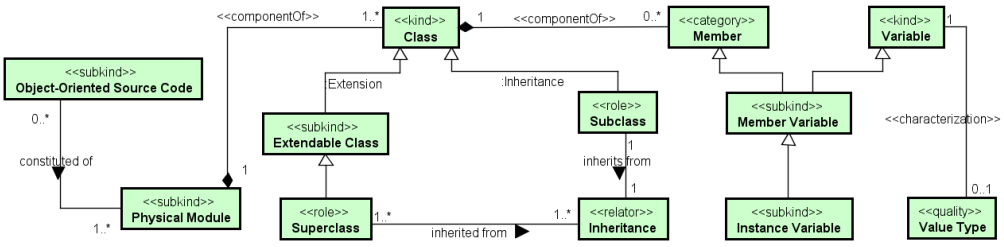

Figura 1. Fragmento da ontologia OOC-O [de Aguiar et al. 2019].

A Figura 2 apresenta o fragmento da ontologia RBDS-O reutilizado em ORM-O. Um **Relational Database System** tem **RDBMS Items** para representar os dados de acordo com o modelo Relacional na forma de **Tables** e definir restrições sobre o modelo na forma de **Constraints**. Uma **Table** é definida por um **Line Type**, cujas linhas possuem um único tipo. Cada **Line Type** é constituído de **Columns**, representando os campos da tabela. Por sua vez, uma **Column** é especificada por uma **Column Constraint**, seja **Column Type Constraint** para restringir os valores da coluna, **Primary Key Constraint**

para definir uma linha única na tabela, ou **Foreign Key Constraint** para referenciar uma determinada chave-primária.

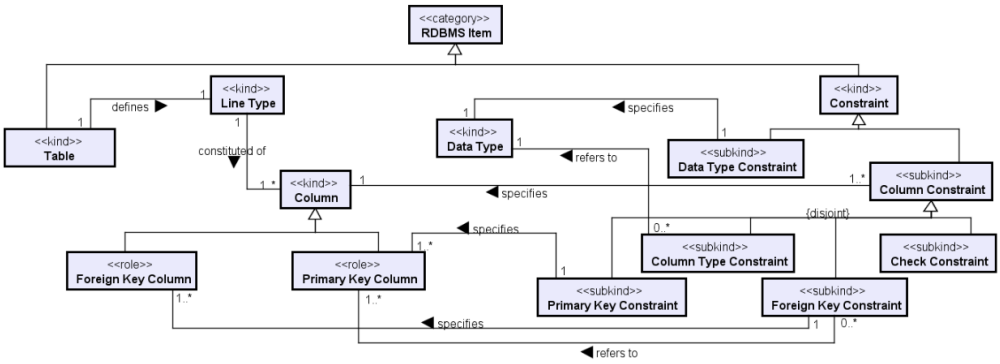

Figura 2. Fragmento da ontologia RBDS-O [de Aguiar et al. 2018].

4. A Ontologia de Frameworks de Mapeamento Objeto/Relacional

A ORM-O (*Object/Relational Mapping Ontology*) é uma ontologia de referência no domínio de *frameworks* ORM. A ontologia visa identificar e representar os conceitos do mapeamento objeto/relacional no escopo de código fonte. Sendo assim, considerando o escopo das aplicações vislumbradas, como a ferramenta de migração que converte código de um *framework* ORM para outro, os padrões de configuração dos *frameworks* com o banco de dados e com o sistema gerenciador de banco de dados não são cobertos pela ORM-O.

A ontologia foi construída aplicando as fases iniciais do método SABiO [Falbo 2014], a saber: Identificação do Propósito e Levantamento de Requisitos, levando à definição de requisitos funcionais e não-funcionais; Captura e Formalização da Ontologia, resultando em sua representação em um modelo gráfico e registrando de forma objetiva a conceituação do domínio; e Implementação, com o uso da linguagem operacional OWL.

Como requisitos funcionais, foram levantadas as seguintes questões de competência (QCs): **CQ01**: Que classes são mapeadas para o banco de dados? **CQ02**: Como os relacionamentos entre classes são mapeados para o banco de dados? **CQ03**: Que atributos de uma dada classe são mapeados para o banco de dados? **CQ04**: Que atributos de uma dada classe são mapeados para chave primária no banco de dados? **CQ05**: Que atributos de uma dada classe são mapeados para chave estrangeira no banco de dados? **CQ06**: Como os relacionamentos de herança entre classes são mapeados para o banco de dados?

Durante a captura e formalização, a ORM-O foi representada em OntoUML [Guizzardi 2005] para análise ontológica à luz da ontologia de fundamentação UFO [Guizzardi and Wagner 2004]. Além disso, para garantir a compreensão consensual do domínio, os conceitos foram definidos em um dicionário de termos e mapeados para os conceitos de cada linguagem de programação com o uso dos *frameworks* selecionados.[2]

[2]O dicionário de termos pode ser visto na especificação completa da ontologia, disponível online em: `https://nemo.inf.ufes.br/projects/sfwon/`.

A captura da ontologia foi apoiada por um processo de aquisição de conhecimento que utilizou documentação de *frameworks* em três diferentes linguagens que utilizam o paradigma orientado a objeto: Java Persistence API (JPA) [JPA 2019], que é o padrão de implementação de *frameworks* para Java, como o Hibernate; Django [Django 2019] e SQLAlchemy [SQLAlchemy 2019] em Python; e QxORM [QxOrm 2019] e ODB [ODB 2019] em C++. A escolha dos *frameworks* para cada linguagem foi feita com base na popularidade de cada um no site *Stack Overflow*,[3] um site de perguntas e respostas amplamente conhecido e utilizado pela comunidade de desenvolvedores. De forma similar, as linguagens também foram escolhidas devido à sua popularidade [de Aguiar et al. 2019].

Com a finalidade de facilitar o entendimento e a apresentação, a ORM-O foi subdividida em 3 partes: **ORM-O Classes and Relationships**, que trata do mapeamento individual das classes e suas relações com outras classes; **ORM-O Inheritance**, que trata do mapeamento de herança e **ORM-O Variable**, que trata do mapeamento de variáveis (atributos). A Figura 3 apresenta a composição da ORM-O. A seguir, serão apresentadas as três sub-ontologias de ORM-O.

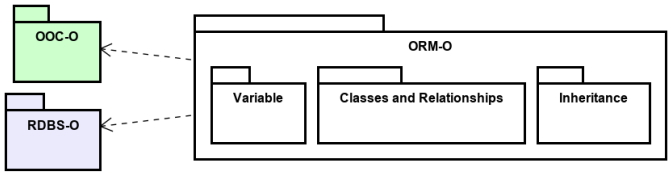

Figura 3. Composição da ORM-O e sua associação com OOC-O e RBDS-O.

4.1. ORM-O Classes and Relationships

A Figura 4 apresenta os conceitos da ORM-O relativos ao mapeamento objeto/relacional de classes e suas possíveis relações. Dado que classes são mapeadas para tabelas, os conceitos **Class** (OOC-O) e **Table** (RDBS-O) são aqui reutilizados.

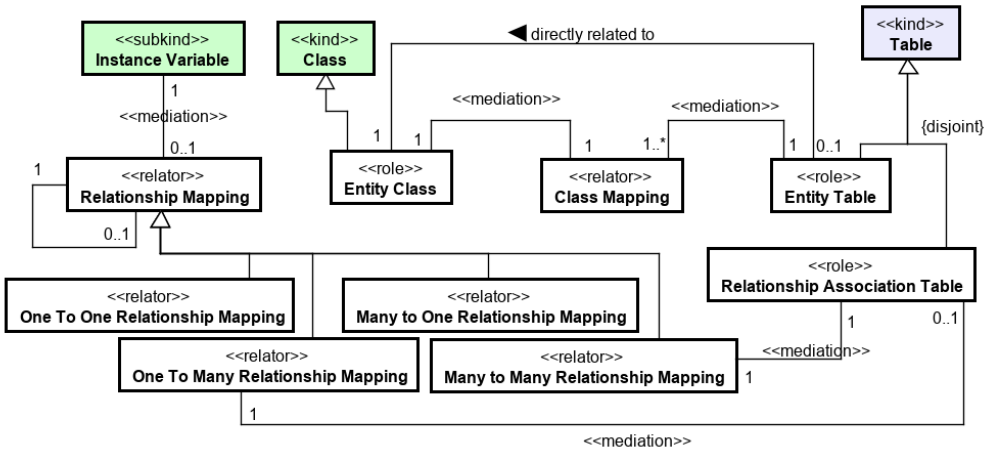

Figura 4. Ontologia de Mapeamento O/R: Classes e Relacionamentos.

O conceito **Entity Class**, que é uma especialização de **Class**, advém da representação de uma classe que será efetivamente mapeada para o banco de dados. Tal

[3] https://stackoverflow.com/

conceito se faz necessário visto que, em um código-fonte, nem todas as classes terão suas instâncias armazenadas no banco de dados.

A formalização do mapeamento de uma **Entity Class** para o banco de dados é feita pela definição do conceito **Class Mapping**, que por sua vez está associado a uma **Entity Table**. Esta última é uma especialização de **Table** para representar tabelas que armazenam tuplas que representam instâncias de classes. Uma **Entity Class** pode ainda estar diretamente relacionada com uma **Entity Table**, indicando assim que determinadas características suas, como o nome por exemplo, são determinadas pela instância de **Entity Class** ligada nessa relação.

O relacionamento entre as classes é mapeado para as tabelas por meio do conceito **Relationship Mapping**. Esse conceito possui quatro diferentes especializações: relacionamento de um para um (**One To One**), um para muitos (**One To Many**), muitos para um (**Many To One**) e muitos para muitos (**Many to Many**). Os conceitos **One To Many Relationship Mapping** e **Many To Many Relationship Mapping** estão associados a uma especialização de **Table**: a **Relationship Association Table**, que é responsável pelo armazenamento de chaves que relacionam as tuplas de cada tabela.

Ainda a respeito de **Relationship Mapping**, é possível que uma instância desse conceito seja associada a outra, do mesmo conceito, representando assim um relacionamento bidirecional.

4.2. ORM-O Inheritance

Os conceitos de mapeamento de herança capturados e formalizados pela ORM-O estão apresentados na Figura 5. Assim como a **Entity Class** especializa **Class**, os conceitos **Subclass** e **Superclass** foram especializados respectivamente em **Entity Subclass** e **Entity Superclass**. Ambos estão associados ao **Inheritance Mapping**.

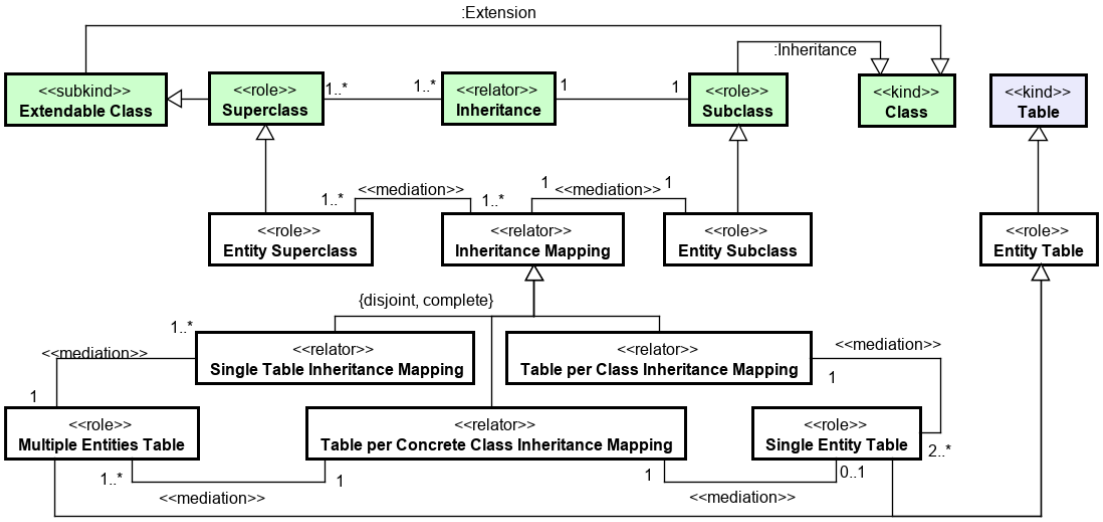

Figura 5. Ontologia de Mapeamento O/R: Herança.

Em mapeamento objeto/relacional, existem três diferentes estratégias de mapeamento de herança [Ambler 2010]: *Single Table*, onde existe uma única tabela para toda

a hierarquia; *Table per Class*, onde cada classe envolvida na hierarquia, seja ela mãe ou filha, é mapeada para uma tabela diferente no banco de dados; e *Table per Concrete Class*, onde cada classe concreta da hierarquia possui sua tabela e nela são incluídas colunas para armazenamento de atributos herdados das classes mães. Sendo assim, especializamos o conceito **Inheritance Mapping** de três maneiras: **Single Table Inheritance Mapping**, **Table per Concrete Class Inheritance Mapping** e **Table per Class Inheritance Mapping**. Os dois primeiros estão associados à **Multiple Entities Table**, uma especialização de **Entity Table** que representa uma tabela do banco de dados relacional que armazena uma ou mais classes diferentes.

Table per Class Inheritance Mapping está associado à **Single Entity Table**, que representa uma tabela do banco de dados que possui informações de uma única classe. Analogamente, há a possibilidade de **Table per Concrete Class** estar associado a uma **Single Entity Table** caso a classe mãe da hierarquia seja concreta, portando há uma associação entre esses dois conceitos.

4.3. ORM-O Variable

O mapeamento de variáveis por meio de *frameworks* ORM tem seus conceitos representados na Figura 6. No diagrama é possível observar o conceito de **Variable Mapping**, que associa um **Mapped Variable**, que é uma especialização de **Instance Variable** que representa uma variável que deve ser mapeada, a uma **Column**. Por sua vez, a possibilidade ou não de mapeamento de valores nulos a colunas do banco de dados é representada pelo conceito **Nullability** que caracteriza uma **Mapped Variable**.

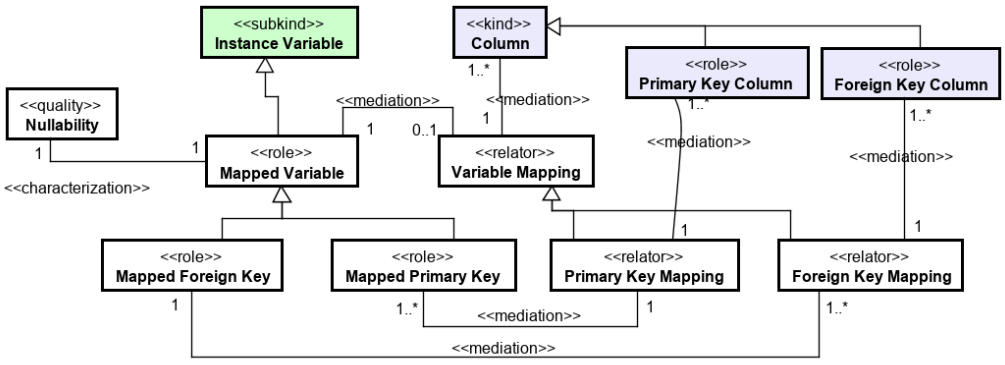

Figura 6. Ontologia de Mapeamento O/R: Variável.

Já uma chave primária é formalizada com a especialização de **Mapped Variable** em **Mapped Primary Key** e seu mapeamento é feito por **Primary Key Mapping**, uma especialização de **Variable Mapping**, que associa ao conceito **Primary Key Column**, especialização de **Column**. Analogamente, **Mapped Foreign Key**, que é uma especialização de **Mapped Variable** para representar o conceito de chave estrangeira, tem seu mapeamento representado por uma especialização de **Variable Mapping** formalizada por **Foreign Key Mapping**.

5. Avaliação

A avaliação da ORM-O foi realizada por meio de atividades de verificação e validação, seguindo o método SABiO [Falbo 2014]. A verificação foi feita por meio da identificação

das respostas às questões de competência utilizando os conceitos que compõem a ontologia. A Tabela 1 apresenta as respostas das QCs levantadas na Seção 4, mostrando quais conceitos e relações são utilizados para responder cada questão.

Tabela 1. Respostas às Questões de Competência.

ID	Resposta
CQ01	**Entity Class** *subtype of* **Class** **Class Mapping** *mapped by* **Entity Class** **Class Mapping** *mapped to* **Entity Table** **Entity Table** *subtype of* **Table**
CQ02	**Instance Variable** *mediation* **Relationship Mapping** **One To One Relationship Mapping**, **One To Many Relationship Mapping**, **Many to One Relationship Mapping** e **Many to Many Relationship Mapping** *subtype of* **Relationship Mapping** **Entity Table** e **Relationship Association Table** *subtype of* **Table** **Many to Many Relationship Mapping** *mediation* **Relationship Association Table** **One to Many Relationship Mapping** *mediation* **Relationship Association Table**
CQ03	**Mapped Variable** *subtype of* **Instance Variable** **Mapped Variable** *mediation* **Variable Mapping** **Variable Mapping** *mediation* **Column**
CQ04	**Mapped Variable** *subtype of* **Instance Variable** **Mapped Primary Key** *subtype of* **Mapped Variable** **Mapped Primary Key** *mediation* **Primary Key Mapping** **Primary Key Mapping** *mediation* **Primary Key Column**
CQ05	**Mapped Variable** *subtype of* **Instance Variable** **Mapped Foreign Key** *subtype of* **Mapped Variable** **Mapped Foreign Key** *mediation* **Foreign Key Mapping** **Foreign Key Mapping** *mediation* **Foreign Key Column**
CQ06	**Entity Subclass** *subtype of* **Subclass** **Entity Subclass** *mediation* **Inheritance Mapping** **Entity Superclass** *subtype of* **Superclass** **Entity Superclass** *mediation* **Inheritance Mapping** **Single Table Inheritance Mapping**, **Table per Class Inheritance Mapping** e **Table per Concrete Class Inheritance Mapping** *subtype of* **Inheritance Mapping** **Entity Table** *subtype of* **Table** **Multiple Entities Table** e **Single Entity Table** *subtype of* **Entity Table** **Single Table Inheritance Mapping** *mediation* **Multiple Entities Table** **Table per Concrete Class Inheritance Mapping** *mediation* **Single Entity Table** **Table per Concrete Class Inheritance Mapping** *mediation* **Multiple Entities Table** **Table per Class Inheritance Mapping** *mediation* **Single Entity Table**

Para a validação da ORM-O, os conceitos da ontologia foram instanciados com o *framework* Hibernate, um dos *frameworks* analisados na atividade de aquisição do co-

nhecimento. A Tabela 2 apresenta alguns desses conceitos com o intuito de validar se a ontologia pode representar situações reais de mapeamento objeto/relacional. A tabela de validação completa pode ser consultada no documento de especificação da ontologia, disponível online em: `https://nemo.inf.ufes.br/projects/sfwon/`.

Tabela 2. Instâncias dos conceitos de ORMF-O.

Conceito	Instância

Entity Class & Class Mapping	`@Entity`
	`public class Person {...};`
	A existência de uma **Entity Class** (Person) faz com que exista implicitamente um **Class Mapping**.

Entity Subclass & Entity Superclass	`@Entity`
	`public class Student extends Person {...};`
	A existência de uma **Entity Subclass** (Student) faz com que a classe que ela estende seja uma **Entity Superclass** (Person).

Table per Class Inheritance Mapping	`@Entity`
	`@Inheritance(strategy = InheritanceType.JOINED)`
	`public class Person {...}`

Entity Table	`@Entity`
	`@Table(name="Person")`
	`public class Person {...}`
	Caso a anotação @Table seja omitida, a **Entity Table** será a tabela do banco de dados cujo o nome é idêntico ao nome da **Entity Class**.

One To Many Relationship Mapping	`@Entity`
	`public class Class {`
	`...`
	`@OneToMany(mappedBy="studentClass")`
	`private List<Student> students;`
	`...`
	`}`

Mapped Variable & Variable Mapping	`@Column(name="enrollment_number")`
	`private Long enrollmentNumber;`
	Qualquer atributo de uma **Entity Class** que não seja precedido da anotação @Transient é uma **Mapped Variable**. A existência de uma **Mapped Variable** implica na existência de uma **Variable Mapping**. Caso a anotação @Column seja omitida, a **Variable Mapping** será associada à **Column** com nome idêntico ao nome da **Mapped Variable**.

Por fim, a ontologia foi utilizada na prática como *interlingua* em uma ferramenta de migração de código desenvolvida a partir da ORM-O. A partir de código Java com mapeamentos objeto/relacional feitos com uso do *framework* ORM Hibernate, a ferramenta gera instâncias dos conceitos da ORM-O (em OWL) e, em seguida, é capaz de gerar o código fonte em Python com uso do *framework* ORM Django. O leitor interessado pode experimentar a ferramenta a partir de seu código-fonte, disponível em `https://github.com/nemo-ufes/ORMFConverter`.

6. Trabalhos Relacionados

Ontologias que envolvem os conceitos pertinentes ao domínio de objeto ou dados relacionais são fundamentadas a partir de diferentes contextos. Observamos ontologias elaboradas para publicação automática de dados semânticos a partir de banco de dados [Trinh et al. 2006], geração de banco de dados a partir de ontologia [de Laborda and Conrad 2005], modelagem de software [Evermann and Wand 2005] e metodologia de desenvolvimento de software [Pastor 1992] apoiados em ontologia de objetos. No entanto, até onde temos conhecimento, poucas pesquisas se aprofundaram na relação existente entre esses domínios, explorada nesse artigo como *frameworks* de linguagem de programação ORM.

[Calero et al. 2006] apresenta uma ontologia para o recurso objeto-relacional do padrão SQL:2003 a fim de clarificar seus elementos e identificar suas inconsistências. A ontologia é subdividida em *DataTypes* (conceitos pertinentes a tipo de dados) e *Schema-Objects* (conceitos pertinentes ao contexto geral, tabelas, restrições e colunas). Diferentemente de ORM-O, o modelo é representado em UML e regras OCL. Ademais, embora o propósito dessa ontologia seja representar aspectos objeto-relacionais de uma esquema de banco de dados, a ontologia não explora essa relação entre os dois domínios e apenas representa os conceitos relacionais na forma de um diagrama de objeto.

[Chang et al. 2001] apresenta a especificação de um metamodelo sobre *data warehouse*. O modelo é dividido, entre outros, em *Object Model* (conceitos base relacionados a classe e objeto) e *Resource* (conceitos relacionados a representação relacional). Diferentemente de ORM-O, o modelo é representado em UML e define relações de especialização entre os domínios de objeto e dados relacionais. Assim, *Column* e *Data Type* do submodelo *Relational* são especializações de *Attribute* e *Classifier* do submodelo *Object Model*, respectivamente. Em ORM-O tais conceitos são ancorados na ontologia de fundamentação UFO e relacionados por meio de mapeamentos na linguagem de programação.

No contexto de migração de código-fonte [Plaisted 2013], aplicação prática utilizada como parte da validação da ORM-O, alguns trabalhos propõem o uso de linguagens intermediárias para operacionalizar a migração. O uso de uma *interlingua* facilita a inclusão de novas linguagens na ferramenta, bastando efetuar o mapeamento entre a linguagem e a *interlingua*, sem necessidade de mapear a nova linguagem a todas as outras já incluídas. Em [Schaub and Malloy 2016], por exemplo, uma linguagem denominada iJava foi mapeada para Java, Python e C++, permitindo migração de código entre estas linguagens de programação. Além de incluir os mapeamentos objeto/relacionais, a conversão feita com base na ORM-O utiliza um modelo conceitual mais bem fundamentado, construido a partir de um processo de Engenharia de Ontologias.

7. Conclusão

Neste artigo, apresentamos uma ontologia de *frameworks* de mapeamento objeto/relacional. A ORM-O foi construída de acordo com um método de engenharia de ontologias e baseada em fontes de amplo conhecimento da comunidade de desenvolvimento de software. Para conceituar o mapeamento objeto/relacional, a ontologia introduz relacionamentos entre as ontologias de bancos de dados relacionais RBDS-O [de Aguiar et al. 2018] e de código orientado a objetos OOC-O [de Aguiar et al. 2019].

As atividades de verificação e validação tiveram resultado positivo alcançado através de respostas que atendiam às questões de competência e apresentando a cobertura dos conceitos fundamentais em um código de linguagem de programação OO utilizando um *framework* ORM bastante popular em Java. Além disso, com o desenvolvimento da ferramenta de migração, foi possível utilizar a ontologia na prática, migrando código de um *framework* ORM (Hibernate em Java) para outro (Django em Python).

Por fim, como trabalhos futuros, há a intenção de inclusão de mais *frameworks* e linguagens de programação na ferramenta de conversão, bem como o uso da ORM-O em outras aplicações práticas na Engenharia de Software como, por exemplo, definição de *smells* arquiteturais em modelos de projeto de sistemas que utilizam *frameworks* ORM. Pretende-se ainda desenvolver ontologias sobre outros tipos de *frameworks*, como controladores frontais (*Model-View-Controller* — MVC), decoradores, de autenticação e autorização, etc., de modo que também possam ser beneficiados pelas ferramentas desenvolvidas neste contexto.

Acknowledgments

O NEMO (`http://nemo.inf.ufes.br`) recebe atualmente apoio das agências de fomento CNPq (processos 407235/2017-5 e 433844/2018-3), CAPES (processo 23038.028816/2016-41), e FAPES (processo 69382549/2015).

Referências

Ambler, S. W. (2010). Mapping objects to relational databases. `https://www.ibm.com/developerworks/library/ws-mapping-to-rdb/`. Acessado em : 05-05-2019.

Bauer, C. and King, G. (2004). *Hibernate in Action*. Manning, 1 edition.

Calero, C., Ruiz, F., Baroni, A., e Abreu, F. B., and Piattini, M. (2006). An ontological approach to describe the SQL: 2003 object-relational features. *Computer Standards & Interfaces*, 28(6):695–713.

Chang, D., Iyengar, S., et al. (2001). Common warehouse metamodel (CWM) specification. *Object Management Group*, 1.

Date, C. J. (2004). *Introdução a sistemas de bancos de dados*. Elsevier Brasil.

de Aguiar, C. Z., Falbo, R. A., and Souza, V. E. S. (2018). Ontological Representation of Relational Databases. In *Proc. of the 11th Seminar on Ontology Research in Brazil (ONTOBRAS 2018)*, pages 140–151, São Paulo, SP, Brazil. CEUR.

de Aguiar, C. Z., Falbo, R. A., and Souza, V. E. S. (2019). OOC-O: a Reference Ontology on Object-Oriented Code. In *Proc. of the 38th International Conference on Conceptual Modeling (ER 2019)*. (in press).

de Laborda, C. P. and Conrad, S. (2005). Relational. OWL: a data and schema representation format based on OWL. In *Proc. of the 2nd Asia-Pacific conference on Conceptual modelling—Volume 43*, pages 89–96. Australian Computer Society, Inc.

Django (2019). Django Documentation. `https://docs.djangoproject.com/en/2.2/`. Acessado em : 05-05-2019.

Evermann, J. and Wand, Y. (2005). Ontology based object-oriented domain modelling: fundamental concepts. *Requirements Engineering*, 10(2):146–160.

Falbo, R. A. (2014). SABiO: Systematic Approach for Building Ontologies. In *Proc. of the 1st Joint Workshop ONTO.COM / ODISE on Ontologies in Conceptual Modeling and Information Systems Engineering*. CEUR.

Guizzardi, G. (2005). *Ontological Foundations for Structural Conceptual Models*. PhD Thesis, University of Twente, The Netherlands.

Guizzardi, G. and Wagner, G. (2004). A Unified Foundational Ontology and some Applications of it in Business Modeling. In *CAiSE Workshops (3)*, pages 129–143.

Ireland, C., Bowers, D., Newton, M., and Waugh, K. (2009). A classification of object-relational impedance mismatch. In *2009 First International Conference on Advances in Databases, Knowledge, and Data Applications*, pages 36–43. IEEE.

JPA (2019). Java Persistence API Documentation. `https://download.oracle.com/otn-pub/jcp/persistence-2_2-mrel-spec/JavaPersistence.pdf`. Acessado em : 05-05-2019.

ODB (2019). ODB Documentation. `https://www.codesynthesis.com/products/odb/doc/manual.xhtml`. Acessado em : 05-05-2019.

Pastor, O. (1992). *Diseño y Desarrollo de un Entorno de Producción Automática de Software basado en el modelo orientado a Objetos*. PhD thesis, Universitat Politècnica de València.

Plaisted, D. A. (2013). Source-to-source translation and software engineering. *Journal of Software Engineering and Applications*, 6(04):30.

QxOrm (2019). QxOrm Documentation. `https://www.qxorm.com/qxorm_en/manual.html`. Acessado em : 05-05-2019.

Schaub, S. and Malloy, B. A. (2016). The Design and Evaluation of an Interoperable Translation System for Object-Oriented Software Reuse. *Journal of Object Technology*, 15(4):1–1.

Souza, V. E. S., Falbo, R. A., and Guizzardi, G. (2009). Designing Web Information Systems for a Framework-based Construction. In Halpin, T., Proper, E., and Krogstie, J., editors, *Innovations in Information Systems Modeling: Methods and Best Practices*, chapter 11, pages 203–237. IGI Global, 1 edition.

SQLAlchemy (2019). SQLAlchemy Documentation. `https://docs.sqlalchemy.org/en/13/`. Acessado em : 05-05-2019.

Teixeira, M. H. T. H. (2017). Impedância objeto relacional — O atrito natural entre os dois mundos. *Tecnologias em Projeção*, 8(1):11–19.

Trinh, Q., Barker, K., and Alhajj, R. (2006). RDB2ONT: A tool for generating OWL ontologies from relational database systems. In *Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services (AICT-ICIW'06)*, pages 170–170. IEEE.