Uso de ontologia no projeto de *software* baseado em *framework*

Luiz Gustavo Dias[1]
Vaston Gonçalves da Costa[2]

Resumo: A gestão do conhecimento é uma disciplina que utiliza diversas ferramentas e técnicas com vistas a auxiliar na produção, gerência e recuperação de produtos e processos como exemplo de ferramenta pode se citar ontologias. Ontologias, podem ser utilizadas para classificar, representar, recuperar e difundir o conhecimento acerca de produtos e serviços. Quando as ferramentas e técnicas da gestão do conhecimento pode ser aplicada na produção de artefato de software, desde o levantamento de requisitos até a realização de testes de qualidade. Paralelo a isto, no que tange o processo de desenvolvimento de *software,* é comum a realização de tarefas genéricas e similares a vários projetos com vias a minimizar o tempo de produção de sistemas por meio de reuso e da adoção de *frameworks*, que podem ser entendidos como um conjunto de classes colaborativas que proporcionam boas práticas, evitando a construção desnecessária de código. Isto posto, no presente artigo apresentam-se resultados sobre a utilização de ontologias de domínio no processo de projeto e implementação de *software web* baseado em *framework*. Foram identificados como principais benefícios da utilização desta técnica a organização dos requisitos do sistema em questão, seus relacionamentos e nível de descrição hierárquica, o que contribui para a identificação de inconsistências em determinados setores do *software*.

Palavras-chave: Framework. Ontologia. Representação do conhecimento.

1 Introdução

A Gestão do Conhecimento (GC) surgiu por volta da década de 1980 e visa aproveitar o capital intelectual das empresas, possibilitando o alinhamento dos objetivos e estratégias, internas e externas.

Voltada para o domínio da tecnologia da informação, as técnicas e ferramentas advindas da GC podem ser utilizadas, por exemplo, no processo de desenvolvimento de *software*, para o gerenciamento de documentos, rastreabilidade de problemas e reutilização de *software*.

Em computação, destaca-se como subárea orientada ao conhecimento, a Engenharia de *Software* (ES), principalmente pelos casos de sucesso estarem altamente ligados às experiências das pessoas que trabalham em projetos, independente do estágio, seja na fase de idealização, desenvolvimento, testes ou implantação. Cada fase da ES é composta por grande número de entidades que segundo Desousa (2003), podem ser entendidas como pessoas, tarefas e artefatos, dentre outros.

Dentre as principais técnicas utilizadas na GC destacam-se o *brainstorming*, onde ideias e intuições são valorizadas e discutidas por grupos heterogêneos, realizando associações de ideias, essa técnica permite romper bloqueios criativos ajudando os envolvidos a pensar fora da caixa; a análise SWOT que consiste na análise integrada dos principais aspectos que caracterizam a posição estratégica da organização em determinado momento; e ontologias, que são definidas por Gruber (1993), como especificação formal e

[1]Universidade Federal de Goiás – UFG. Regional Catalão, Mestrado Profissional em Gestão Organizacional. Contato: gusttavodiias@gmail.com
[2]Universidade Federal de Goiás – UFG. Regional Catalão, Unidade Acadêmica Especial de Matemática e Tecnologia. Contato: vaston@ufg.br.

explícita de uma conceitualização compartilhada, que apoiam a organização nos processos de classificação, representação e recuperação do conhecimento.

Será utilizada nesse trabalho a técnica de ontologia para projeção do produto de *software*, tendo em vista a interdisciplinaridade e a complexidade de aplicação.

Desta forma, este artigo tem a finalidade mostrar resultados acerca de um projeto que se encontra em fase de teste, realizado em uma unidade acadêmica, onde um produto de *software* foi projetado baseado em ontologia, implementado e testado utilizando um *framework* para desenvolvimento web.

A questão que guiou a pesquisa foi: existem benefícios em se utilizar uma técnica da gestão do conhecimento para projetar um produto de *software* baseado em *framework*? A resposta à pergunta norteadora se encontra na seção de conclusão.

Nas seções seguintes apresentam-se o referencial teórico sobre gestão do conhecimento, desenvolvimento de *software* baseado em *framework*, seguida pelos procedimentos utilizados para a realização da pesquisa, os resultados obtidos e as considerações finais.

2 Gestão do Conhecimento

Segundo Terra (2001), a GC está relacionada à aptidão das organizações em conseguir captar e utilizar o conhecimento organizacional que auxilie no desenvolvimento de processos ou produtos.

Desta maneira tem-se que, a GC utiliza técnicas e ferramentas para captar diferentes tipos de conhecimento organizacional, que desenvolvam capacidades específicas e de inovação, e para que isso se torne possível, é necessário gerenciar, produzir, organizar e compartilhar o conhecimento por toda a organização.

Dentro da GC destacam-se as ontologias, que segundo Isotani e Bittencourt (2005), tem o termo derivado da filosofia onde "onto" significa ser, e "logia" significa conhecimento, logo a palavra ontologia significa estudo e conhecimento das coisas, e consiste em esclarecer o ser. De forma geral, pode-se entender por ontologia, uma descrição formal de determinado domínio, onde se estruturam relacionamentos com base nos significados, permitindo que o vocabulário organizado nessa estrutura seja reutilizado. Em computação por exemplo, a formalização de conceitos por meio de ontologias se torna fundamental pois permite a legibilidade por parte das máquinas (GUIMARÃES; CALDEIRA; QUARESMA, 2014).

De maneira mais simples, ontologias podem ser definidas como organogramas complexos responsáveis por descrever formalmente um domínio composto por conceitos, propriedades de cada classe, e restrições, possibilitando visão macro do domínio em questão, onde todos os termos pertencentes a ele são relacionados e caracterizados.

Os propósitos básicos de uma ontologia são: armazenar uma taxonomia para a representação de entidades, que neste caso podem ser entendidas como requisitos, métodos, objetos e afins; segundo, armazenar as formas lógicas que conectam as entidades de acordo com as regras de negócio relacionadas ao

sistema; terceiro, armazena axiomas que podem ser definidos fórmulas lógicas que servem para moldar restrições referentes a instâncias definidas a partir de entidades.

Ontologias são utilizadas em diferentes domínios, que podem variar desde a área da saúde quanto a antropologia por exemplo. Silva et al. (2007) propuseram uma ontologia voltada para segurança da Informação tendo como base para sua criação normas e políticas de segurança definidas por consórcios e ISOS, Dias e Costa (2016) propuseram uma ontologia para gerência de habilidades no âmbito organizacional e aplicaram o modelo proposto a consultas desenvolvidas em *desciption logics*.

Levando em consideração os diferentes domínios de aplicação, existem diversos tipos de ontologia, que são mais adequadas para resolver determinadas questões, Isotani e Bittencourt (2015) citam em seu trabalho, diferentes tipos de ontologias, destacando-se para aplicação nesta pesquisa a ontologia de domínio.

As ontologias de domínio definem determinado contexto bem como atividades e atores relacionados a ele, define um vocabulário voltado ao compartilhamento dentro de um domínio específico cujos principais objetivos são o compartilhamento e o reuso.

3 Desenvolvimento de *Software* baseado em *Framework*

Um dos objetivos da ES é que códigos sejam reusados durante o processo de desenvolvimento de *software*, pois assim, segundo Gimenez e Huzita (2005), alcança-se maior qualidade e menor esforço durante o desenvolvimento.

Baseado nisso, tem-se o desenvolvimento de software baseado em *framework* que possibilita o desenvolvimento de vários produtos similares, a partir de uma mesma estrutura, que contém soluções para os problemas gerais nas aplicações.

Existem inúmeras definições para *framework* na literatura, Fayad et al (1999) dizem que um *framework* é formado por uma quantidade determinada de classes para a solução de uma quantidade de problemas. Para Mattson (1999) um framework possui a finalidade de reutilizar o máximo possível, representado por diferentes tipos de classes em potencial para especialização.

Além da necessidade em reuso de código-fonte, a demanda por aplicações multiplataforma vem crescendo dia após dia, logo, é de extrema importância que tais aplicações sejam robustas, escaláveis e otimizadas, tendo em vista o público heterogêneo que fará uso das mesmas. Desta forma, cresceu consideravelmente a quantidade de *frameworks* para o desenvolvimento de *software* para a *web.* Desta forma, as aplicações são desenvolvidas com perfil multiplataforma, além de fazer uso de boas práticas, priorizando o reuso, a qualidade e o tempo de desenvolvimento.

Os principais *frameworks web* disponíveis no mercado podem ser classificados levando em consideração várias características, principalmente quanto a curva de aprendizado. Pois, quanto mais robusto, maior tempo é demandado ao programador para que o mesmo domine a ferramenta. Para o desenvolvimento desta pesquisa foi utilizado na fase de implementação o

framework Laravel[3], desenvolvido por Taylor B. Otwell em 2011. As razões para escolha do mesmo foram principalmente por ser modular, conter vasta documentação online, pelo perfil cooperativo da comunidade utilizadora, pela curva de aprendizado que é consideravelmente menor quando comparado à outras ferramentas de mesmo porte e também por sua arquitetura, tendo em vista que Laravel faz uso da arquitetura MVC (*model, view*, *controller*).

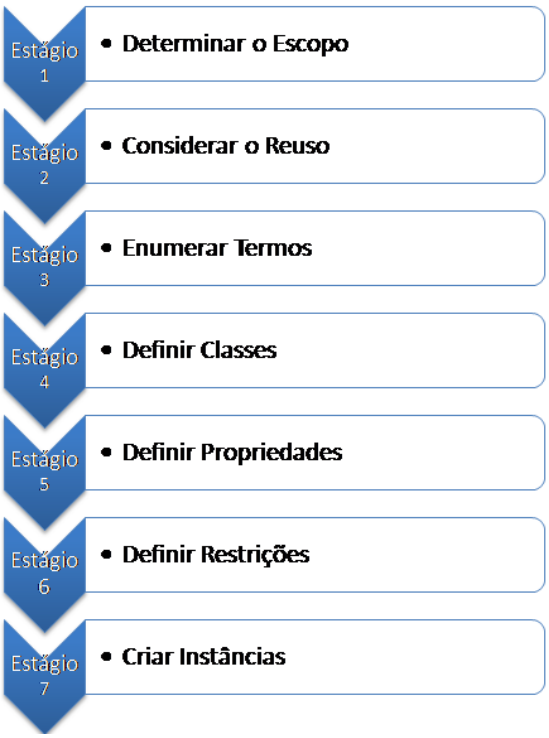
4 Método

Este estudo possui abordagem qualitativa, uma vez que um produto de *software* está sendo desenvolvido para automatizar o processo de solicitar defesas de projetos de pesquisa em uma unidade acadêmica, e não serão utilizadas técnicas estatísticas para a análise dos dados.

As respostas serão buscadas sem a necessidade da comprovação numérica, preocupando-se com aspectos da realidade que não podem ser quantificados, levando em consideração seu caráter exploratório e aspectos específicos do domínio (DALFOVO; LANA; SILVEIRA, 2008; GERHARDT; SILVEIRA, 2009).

Para a construção da ontologia foi utilizada a *Ontology Development 101* (NOY, 2001), e seus estágios são relacionados na figura 1.

Figura 1: Passos da metodologia 101



Fonte: Adaptado de (NOY, 2001).

[3] https://laravel.com/

O primeiro passo da metodologia selecionada foi determinar o escopo de, onde o mesmo foi o desenvolvimento de *software* baseado em *framework*; o segundo estágio remete à possibilidade em se utilizar ontologias já existentes; no terceiro passo são listados o máximo de termos do domínio e enumerá-los em ordem de significância; o quarto estágio é relacionado à classificação dos termos listados no passo anterior, onde são agrupados termos que possuem semelhanças; o quinto passo é relacionado à levantar as propriedades de classes específicas; no sexto estágio são definidas as restrições entre as classes, que podem ser definidas por exemplo no modelo de hierarquia; e por fim o último passo é voltado à criação de instâncias que podem ser entendidas como indivíduos do domínio. No caso dessa pesquisa, os indivíduos considerados foram os métodos utilizados no desenvolvimento do sistema, os *models* e as *views*.

Para o desenvolvimento da ontologia, foi utilizado o editor de ontologias Protegé[4] em sua versão online. Para o desenvolvimento do sistema, foi utilizado o *framework* Laravel[5], que faz uso da linguagem php como linguagem de programação principal, é utilizado o paradigma orientado a objetos, utiliza a arquitetura MVC (*model, view, controller*) que possibilita principalmente o reuso de código e a modularização de componentes. Como ferramentas *case* foram utilizadas como IDE o editor ATOM, o editor de diagramas DIA, o servidor local WAMPP e o editor de texto Word 2017.

5 Resultados e Discussão

Antes de adentrar à fase de projeto do sistema e o uso de ontologia, é necessário entender a finalidade do *software* bem como seu contexto de aplicação. O sistema tem a finalidade de receber informações advindas de alunos e orientadores que pertencem a determinada unidade acadêmica, acerca de projetos desenvolvidos pelos mesmos, para que seja criado a partir daí todo documental necessário relacionado à tramites administrativos, como declaração de solicitação de defesa de projeto, declaração de participação de bancas, ata de defesa, dentre outros.

Durante a idealização do mesmo, percebeu-se considerável número de regras de negócio envolvendo principalmente níveis de acesso à informação, tendo em vista que a solicitação só é de fato efetivada, após o usuário aluno realizar sua solicitação e a mesma for aprovada pelo usuário orientador, desta forma nota-se que o sistema possui três diferentes tipos de usuários com privilégios diferentes, para isso optou-se por trabalhar uma ontologia na fase de projeto.

A ontologia contou com sete classes principais, voltadas a categorizar todos os artefatos relacionados ao sistema idealizado, o que pode ser visualizado na Figura 2. Optou-se por estrutura-la levando em consideração a arquitetura utilizada pelo *framework* Laravel, ou seja, a arquitetura MVC.

[4] http://webprotege.stanford.edu
[5] https://laravel.com/

Figura 2: Estrutura do *software* segundo a arquitetura utilizada pelo *framework*.

Fonte: Os autores.

O Sistema foi projetado levando em consideração três superclasses básicas, *Controller*, responsável por direcionar as requisições advindas dos usuários ao *Model*, que pode ser entendido como objetos de sistema à *View*, que são as telas do sistema. A utilização dessa arquitetura, faz com que o *software* seja modular, possibilitando com que a equipe de desenvolvimento, trabalhe simultaneamente sem impactar módulos já desenvolvidos, desta forma a ordem do desenvolvimento do código não é considerado fator de risco.

Além das classes principais (*Model*, *View* e *Controller*), a ontologia foi composta por subclasses AdminController, DefesaController, destinadas a classificar determinados indivíduos, como mostrado na Figura 3.

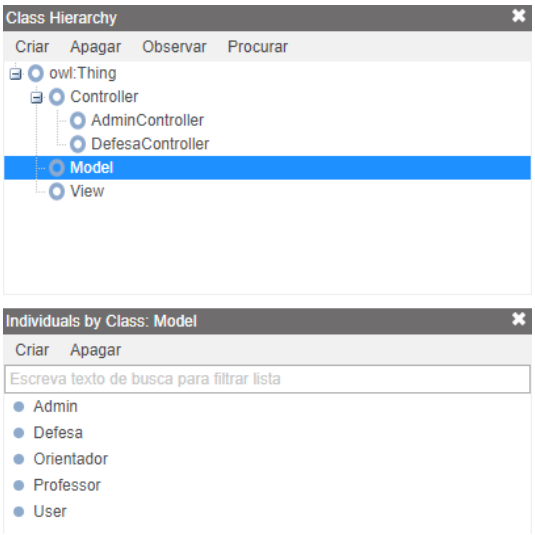
Figura 3: Relação entre classes e indivíduos da ontologia.

Fonte: Os autores.

Como percebido na Figura 3, as regras de negócio e níveis de acesso relacionadas ao usuário administrador, foram centralizadas em um controlador específico, denominado AdminController. De forma análoga, as regras de negócio relacionadas construção de defesas, foram centralizadas em um controlador específico, denominado DefesaController.

O processo de classificar indivíduos foi utilizado para as outras duas superclasses, onde na classe *model* foram agrupados os objetos do sistema, e *view* as telas de todos os usuários e níveis de acesso, o que pode ser visualizado nas figuras 4 e 5.

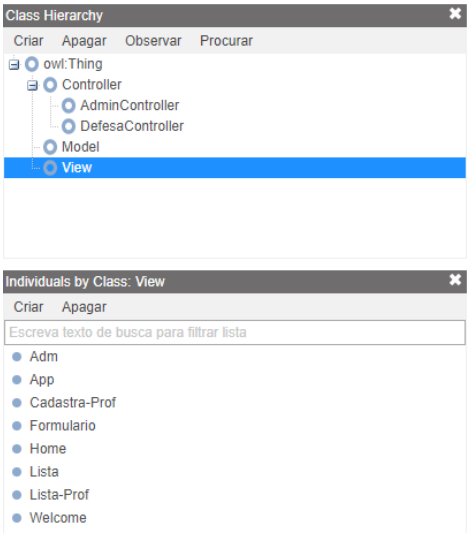
FIGURA 4: Relação dos objetos do sistema.



Fonte: Os autores.

Os objetos do sistema neste contexto, dizem respeito aos componentes envolvidos no domínio, sendo neste caso o administrador do sistema – model admin -, os usuários do sistema – modelos User e Orientador, os professores envolvidos em projetos – model Professor –, e os projetos a serem defendidos - model Defesa.

FIGURA 4: Relação das telas do sistema.



Fonte: Os autores.

Diferente das regras de negócio, as telas não precisam ser classificadas levando em consideração os níveis de acesso, pois quem possui a função de respostas a requisições de usuários são os *controllers,* desta forma, as *views*, estão relacionadas ao layout das telas da aplicação, como disponibilização de formulários, disponibilização de conteúdo em forma de tabelas e etc.

Após desenvolver a ontologia baseada na arquitetura MVC, percebeu-se que além de visão macro dos relacionamentos entre os módulos do sistema, é

possível aplicar testes à ontologia, utilizando um raciocinador lógico como o Pellet[6], para que através de consultas elaboradas utilizando lógica descritiva, sejam encontrados possíveis erros, ambiguidades ou falhas no *software* implementado.

Além disso percebeu-se que, por possibilitar a descrição das propriedades de classes e indivíduos, é possível testar a corretude dos métodos categorizados, em relação à composição da tríade MVC, assim é possível verificar se determinado método é compatível com o *controller* ao qual faz parte, bem como o *model* e sua *view* relacionada.

Percebeu-se também que a desvantagem da utilização de ontologia para as fases de projeto e verificação da corretude, está relacionada a complexidade e dimensão do projeto em questão. Tal processo pode demandar investimento em tempo tanto na fase de projeto quanto na fase de testes, desta forma pode não ser indicado à projetos de pequeno porte. Entretanto no que diz respeito à projetos com maior nível de complexidade, que fazem o uso de diferentes tipos de módulos, como sistemas de transações monetárias por exemplo, a utilização da GC e ontologia nas fases de projeto e testes pode trazer bons resultados, tendo em vista os impactos negativos que erros e inconsistências podem trazer ao usuário do produto final.

Considerações finais

Foi relatada neste trabalho a construção de uma ontologia para a fase de projeto de *software* que auxiliou a fase de implementação. Os resultados preliminares obtidos, apontam que a mesma possui capacidade de auxiliar também no processo de testes tanto de versão, quanto de componentes, pois a metodologia de desenvolvimento da mesma possibilita o uso de raciocinadores lógicos que identificam inconsistências através de propriedades, restrições e relações entre classes de indivíduos, através de inferências.

A próxima etapa consiste em submeter a ontologia à um raciocinador e aplicar regras de inferência de conhecimento, provando assim a viabilidade do uso da gestão do conhecimento por meio de ontologias em diferentes fases do processo de desenvolvimento de *software.* Durante o desenvolvimento da ontologia utilizando o editor de ontologias Protegé em sua versão online, foi notada certa limitação relacionada às funcionalidades, principalmente no que diz respeito ao uso de raciocinadores, logo, para dar continuidade ao projeto, será utilizada a versão do editor para *desktop* em etapas posteriores*.*

[6] https://github.com/stardog-union/pellet

Referências

DALFOVO, M. S.; LANA, R. A.; SILVEIRA, A. Métodos quantitativos e qualitativos: um resgate teórico. *Revista Interdisciplinar Científica Aplicada*, v. 2, n. 3, p. 1–13, 2008. Disponívem em: <http://rica.unibes.com.br/index.php/rica/article/view/243/234 >

DESOUZA, Kevin C. Barriers to Effective Use of Knowledge Management Systems in Software Engineering. Communications of the ACM, vol. 46, n. 1, p. 99-101, jan. 2003. Disponível em: <https://pdfs.semanticscholar.org/0d94/52b54e1226157639d28c40f12441e2f9d32a.pdf>

DIAS, L. G; COSTA, V. G. Emprego de análise formal no processo de gerenciamento de habilidades. Tecnologias em pesquisa: ciência exatas e biológicas. P. 147-154, 2016. Disponível em: <http://pdf.blucher.com.br.s3-sa-east-1.amazonaws.com/openaccess/9788580392326/10.pdf>

Fayad, M. E., Schimidt, D. C. and Johnson, R. (1999) Building application frameworks: Object-oriented foundations of framework design. John Wiley & Sons. 1999.

GERHARDT, T. E.; SILVEIRA, D. T. Métodos de pesquisa. coordenado pela universidade aberta do brasil–uab/ufrgs e pelo curso de graduação tecnológica–planejamento e gestão para o desenvolvimento rural da sead/ufrgs. *Porto Alegre: Editora da UFRGS*, 2009.

GIMENES, I. M. de S.; HUZITA, E. H. M. Desenvolvimento baseado em componentes: conceitos e técnicas. Rio de Janeiro: Ciência Moderna, 2005.

GUIMARÃES, F.; CALDEIRA, C.; QUARESMA, P. Enterprise intelligence com metadatos como ontologia. Escola de Ciências e Tecnologia da Universidade de Évora, 2014. Disponível em: <http://dspace.uevora.pt/rdpc/handle/10174/14067>

GRUBER, T. R. A translation approach to portable ontologies. Knowledge Acquisition, v. 5, n. 2, p. 199-220, 1993. Disponível em: < http://www.sciencedirect.com/science/article/pii/S1042814383710083>

ISOTANI, S.; BITTENCOURT, I. I. Dados Abertos Conectados. São Paulo: Novatec Editora, 2015.

MATTSSON, M. "Object-Oriented Frameworks: A Survey of Methodological Issues", M.Sc. Dissertation, Department of Computer Science and Business Administration, University College of Karlskrona/Ronneby, LU-CS-96-197, 1996.

NOY, N. F. *Ontology Development 101: A Guide to Creating Your First Ontology: Knowldege Systems Laboratory*. Stanford University, 2001. Disponível em: < https://protege.stanford.edu/publications/ontology_development/ontology101.pdf>

SILVA, Geiza et al. Dealing with the formal analysis of information security policies through ontologies: A case study. In: MEYER, Thomas; NAYAK, Abhaya (Ed.). *3rd Australasian Ontology Workshop. (AOW2007)*. Queensland, Australia: ACS, 2007. v. 85. Disponível vem: < https://dl.acm.org/citation.cfm?id=2449265>

TERRA, J.C.; KRUGLIANKAS, I. Gestão do Conhecimento em Pequenas e Médias Empresas. São Paulo: Negócio Editora, 2003.