**ARTIGO ATENÇÃO MUDAR O TÍTULO**

**UM COMPARATIVO DAS METODOLOGIAS ÁGEIS XP E SCRUM PARA O DESENVOLVIMENTO RAD**

Lucas Fagundes Teixeira, Wagner Rodrigo da Silva,

Curso de Bacharelado em Engenharia de Software – Centro de Ensino Superior de Juiz de Fora (CESJF) – Campus Academia  
360100-000 – Juiz de Fora – MG– Brasil

lucasft\_01@outlook.com, [wagnerrodrigo.pan@gmail.com](mailto:wagnerrodrigo.pan@gmail.com)

**1 INTRODUÇÃO**

As metodologias ágeis surgiram para suprir certos entraves no desenvolvimento que as metodologias tradicionais não conseguiam tratar. Feedback rápido com o cliente e rápida adaptação com os inevitáveis problemas ao longo do projeto são apenas algumas de outras complicações não tratadas pelo sistema arcaico. Os resultados dessas dificuldades podem ser notados em alguns dados recolhidos em 1995[1] onde, usando como base 8380 projetos, mostram que apenas 16,2% dos projetos foram entregues respeitando os prazos e os custos e com todas as funcionalidades especificadas. Aproximadamente 31% dos projetos foram cancelados antes de estarem completos e 52,7% foram entregues, porém com prazos maiores, custos maiores ou com menos funcionalidades do que especificado no início do projeto.

A idéia das metodologias ágeis é o enfoque nas pessoas e não em processos ou algoritmos. Além disso, existe a preocupação de gastar menos tempo com documentação e mais com a implementação. Uma característica das metodologias ágeis é que elas são adaptativas ao invés de serem preditivas. Com isso, elas se adaptam a novos fatores decorrentes do desenvolvimento do projeto, ao invés de procurar analisar previamente tudo o que pode acontecer no decorrer do desenvolvimento.

Dentre as várias metodologias ágeis existentes, as mais conhecidas são a *Extreme Programming* e a *Scrum*. E são nestas que o artigo tem enfoque.

# **2 REFERENCIAL TEÓRICO**

Devido a problemas ocorrido da na década de 70 com os processos de desenvolvimento de software os quais não possuía uma metodologia ou um padrão a ser seguido no decorrer da criação das aplicações, como a demanda estava cada vez maior na procurar para criação de sistemas para empresa, a demanda era muito maior que as fabricas de software poderia criar, esse período ficou conhecido como a Crise do Software. Devido os fatos ocorrido no passado foi criada um conjunto de regras pensadas e elaboradas com um propósito de melhoria essa é a proposta de desenvolvimento de software, está metodologia de desenvolvimento foi sugerida por (SOMMERVILLE, 2007).

Com o Manifesto Ágil, foi definido alguns princípios citados a seguir:

1. Nossa maior prioridade é satisfazer os cliente, através da entrega adiantado e continua de software de valor.
2. Aceitar mudanças de requisitos, mesmo no fim do desenvolvimento. Processos ágeis se adequam a mudanças, para que o cliente possa tirar vantagem competitivas.
3. Entregar software funcionando com frequência, na escala de semanas até meses, com preferência aos períodos mais curtos.
4. Pessoas relacionadas à negócios e desenvolvedores devem trabalhar em conjunto e diariamente, durante todo o curso de projeto.

por Beck et al. (2001)

Estas metodologias foram criada para resolver dificuldades encontradas na área de software

# **3. METODOLOGIAS ÁGEIS XP E SCRUM**

## 

## **3.1 XP**

A Extreme Programming (XP) é uma metodologia ágil para equipes pequenas e médias que desenvolvem software baseado em requisitos vagos e que se modificam rapidamente. [2]

Conforme Tabela 1, a metodologia XP, busca trazer valores e motivações para toda equipe envolvida no projeto de desenvolvimento. Levar confiança ao cliente de que suas necessidades serão respeitadas diante de um informativo de prioridades é de vital importância para que o projeto não seja interrompido.

**Tabela 1 – Valores de Extreme Programming**

|  |  |
| --- | --- |
| FeedBack | Comunicação |
| Simplicidade | Coragem |

Mostradas como uma série de atividades a serem seguidas, as práticas de XP norteiam a equipe desenvolvedora durante o projeto. Os valores citados anteriormente, somados as práticas ( Tabela 2), resultam em um grupo de boas atitudes.

**Tabela 2 – Práticas de Extreme Programming**

|  |  |  |
| --- | --- | --- |
| Cliente Presente | Jogo do Planejamento | Integração Contínua |
| Pequenas Versões | Metáforas | Desenvolvimento Orientado a Testes |
| Projeto Simples | Time Coeso | Refatoração |
| Testes de Aceitação | Ritmo Sustentável | Padrões de Codificação |
| Reuniões em Pé | Posso Coletiva | Programação em Pares |

**3.1.1 A Equipe e Seus Membros**

Dentro de uma equipe XP, não existem pessoas com ações mais importantes que as outras, não existindo hierarquia. Cada membro faz sua contribuição ao projeto da melhor maneira que for possível.

**Tabela 3 – Membros de Equipe de Extreme Programming**

|  |  |
| --- | --- |
| Gerente de Projetos | Treinador |
| Analista de Testes | Redator Técnico |
| Desenvolvedor | Rastreador |

**a)** Gerente de Projetos:

Objetivo: Assegurar uma comunicação eficiente entre o cliente e todos os envolvidos no projeto. Também devem monitorar o progresso da equipe e ajudá-la a perceber tudo que foi conquistado.

**b)** Analista de testes:

Objetivo: Ajudar clientes e desenvolvedores a escreverem testes para as histórias. Além disso, auxiliam os desenvolvedores na automatização do testes ao longo da iteração. Quando a equipe não consegue automatizar alguns testes, os analistas de teste os executam manualmente.[4]

**c)** Desenvolvedor:

Objetivo: Ajudar a estimar as histórias, implementar as histórias e automatizar tarefas repetitivas. Também são responsáveis por criarem testes automatizados para tudo o que produzem usando o TDD- que em portuguuês significa desenvolvimento orientado a testes. Além de criarem novas funcionalidades, é necessário refatorar o sistema constantemente para aprimorar a arquitetura, eliminar duplicações e tornar o código mais limpo.[5]

**d)** Treinador:

Objetivo: Deve identificar no que cada um é bom, chamar atenção para oportunidades de melhorias e lembrar a equipe das regras. Pode, eventualmente, fazer programação pareada.[6]

**e)** Redator técnico:

Objetivo: Ajudar a equipe a criar e manter a documentação do projeto. Por tanto, Os redatores técnicos devem compreender quem é o público alvo das publicações, assegurar que a linguagem certa seja usada e fazer com que a informação seja atualizada continuamente.[6]

f) Rastreador:

Objetivo: Coletar as métricas que vão sendo desenvolvidas e compara-las com as métricas que foram estimadas, realizando uma análise das eventuais divergências. Também é responsável por avaliar a viabilidade do objetivo conforme as limitações de recursos e tempo.[6]

**3.1.2 Ciclo de Vida**

Um projeto XP passa pelas seguintes fases: exploração, planejamento inicial, iterações do release, produção, manutenção e morte.[3]

**a)**Exploração:

*Objetivo:* Investigar possíveis soluções e verificar a viabilidade de tais soluções.

*Funcionamento:* Os programadores elaboram possíveis arquiteturas e tentam visualizar como o sistema funcionará considerando o ambiente tecnológico (hardware, rede, software, performance, tráfego) onde o sistema irá rodar.

b)Planejamento inicial:

Objetivo**:** Entrar em acordo com o cliente em uma data de entrega para o primeiro release.

*Funcionamento:* Os programadores, juntamente com o cliente, definem as estórias (use case simplificados) a serem implementadas e as descrevem em cartões. Os programadores assinalam uma certa dificuldade para cada estória e, baseados na sua velocidade de implementação, dizem quantas estórias podem implementar em uma iteração. Depois, os clientes escolhem as estórias de maior valor para serem implementadas na iteração – isso é chamado planejamento de iteração. O processo então se repete até terminar as iterações do release. O tempo para cada iteração deve ser de uma a três semanas e para cada release de dois a quatro meses.

c) Iterações do release:

*Objetivo:* Ter uma idéia melhor das tecnologias e do domínio do problema de modo que as iterações poderão ser mais curtas nos releases subseqüentes e fazer estimativas mais confiáveis com o que se aprendeu das iterações passadas.

*Funcionamento: S*ão escritos os casos de teste funcionais e de unidade. Os programadores vão seguindo mais ou menos o seguinte fluxo de atividades na seguinte ordem (Em cada iteração): escrita dos casos de testes; projeto e refatoramento; codificação; realização dos testes; e integração. À medida que esse fluxo vai sendo seguido, o sistema vai sendo construído segundo os princípios, valores e práticas apresentados nas seções anteriores.

d) Manutenção:

*Objetivo:* Reajustar o processo para conseguir adequar às imprevisibilidades que vão surgindo no decorrer do processo. Exemplo: implementação de novas funcionalidade, chegada de novos membros e a reescrita do código para o seu melhoramento.

*Funcionamento:* Estando em XP sempre simultaneamente produzindo novas funcionalidades, mantendo o sistema existente rodando, incorporando novas pessoas na equipe e melhorando o código. Mecanismos como: refatoramento, introdução de novas tecnologias, e introdução de novas idéias de arquitetura podem ser utilizados em um projeto XP.

e) Morte:

*Objetivo:* Dar fim à um projeto, seja pelo motivo do cliente já estar satisfeito, seja pela inviabilidade em dar continuidade ao desenvolvimento.

*Funcionamento: Quando não mais existir novas histórias, é o momento de finalizar o projeto. É o momento de escrever algumas páginas (de 5 a 10 páginas) sobre a funcionalidade do sistema, um documento que auxilie no futuro a saber como realizar alguma alteração no sistema. Toda a equipe que trabalhou no sistema deve ser reunida para reavaliação. Aproveite a oportunidade de analisar o que pode ter causado queda no sistema e o que fez o projeto avançar.*

Figura 1: Ciclo de Vida do XP

A close up of text on a white background

Description generated with high confidence

Figura 1 retirada do endereço http://www.extremeprogramming.org/map/project.html

## **3.2 Scrum**

Esta metodologia ágil foi criada em 1993 por Jeff Sutherland e, de acordo com o próprio criador, teve como base várias ideias de um estudo conduzido em 1986 por dois professores de administração japoneses, Hirotaka Takeuchi e Ikujiro Nonaka, publicado na Harvard Business Review.

Neste estudo, os dois professores conseguiram identificar que nas empresas haviam pequenas equipes com grande autonomia e liberdade trabalhando movidas por um grande propósito. Os superiores não ditavam como os trabalhos deviam ser feitos, eles apenas removiam os problemas que impediam a equipe de ser bem sucedida. Isto sendo feito, o trabalho era conduzido em um fluxo contínuo de geração de valor, evitando desperdícios de tempo e dinheiro.

Em seu livro, Scrum - a arte de fazer o dobro do trabalho na metade do tempo, Jeff Sutherland fala que para o Scrum realmente funcionar, alguém da equipe sênior de gerenciamento deve ter uma alta compreensão do quão prejudiciais ao fluxo do desenvolvimento os obstáculos são. Além disto, todos devem ter ciência da drasticidade do efeito da eliminação destes obstáculos. No entanto, as pessoas não costumam adotar estas recomendações por exigir muita honestidade consigo mesmas e com os outros.

Apenas com uma breve introdução ao Scrum, já é possível notar a importância dada as pessoas por esta metodologia. Pois, em meio a infinidade de contratempos em um processo e a natural diferença entre os membros do grupo, a melhor alternativa é deixar com que ocorra uma auto-organização e adaptação quando necessário. Mas, para que isso seja possível de ser feito, um conjunto de valores, princípios e práticas devem ser adotados e adequados a realidade da organização. Por este motivo, o Scrum é considerado como um framework altamente adaptável as mais diversas situações e no fim o resultado será uma versão que é exclusivamente sua.

Ao decorrer deste tópico 3.2 será citado as práticas, os valores, membros da equipe e todo o ciclo do Scrum.

**3.2.1 Práticas do Scrum**



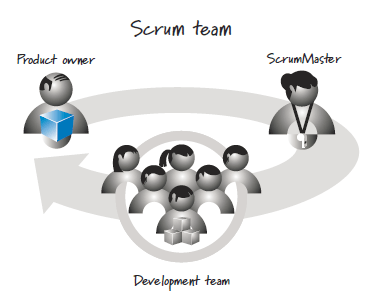
Figura 2: Práticas do Scrum

Figura 2 retirada do endereço <http://www.mindmaster.com.br/scrum/>

Esta imagem demonstra de forma clara todas as práticas do Scrum que serão explicadas logo abaixo.

**3.2.1.1 Papéis Fundamentais**

Figure 3 Time Scrum

*****Figura 3 retirada do livro Essencial Scrum: A Practical Guide to the Most Popular Agile Process*

1. **Product Owner (Dono do Produto)**

Responsável pela visão do que você vai fazer ou conseguir. Ela leva em consideração os riscos e os benefícios, o que é possível, o que pode ser feito e o que desperta a paixão na equipe.

1. **Scrum Master (Mestre Scrum)**

Responsável por orientar toda a equipe em relação à estrutura do Scrum, além de ajudar a eliminar qualquer obstáculo que os esteja deixando mais lentos.

1. **Development Team (Time de Desenvolvimento)**

São quem realmente trabalharão no projeto. Essa equipe precisa ter todas as habilidades necessárias para pegar a visão do Dono do Produto e transformá-la em realidade. Segundo Jeff Sutherland, as equipes devem ser pequenas; entre três e nove pessoas é o princípio básico.

**3.2.1.2 Atividades Básicas e Artefatos**

Figura 4: Processo Scrum

*A close up of text on a white background

Description generated with high confidenceFigura 4 retirada do endereço http://www.mindmaster.com.br/scrum/*

O [Product Owner](http://www.mindmaster.com.br/product-owner/) tem uma visão do produto ou funcionalidade a ser concluída. Como este objetivo a ser concluído pode ser grande, por meio de uma atividade chamada *Grooming,*ele é dividido em um conjunto de funcionalidades que são compilados em uma única lista priorizada chamado de Product Backlog.

Então é feito a primeira reunião de Planejamento de Sprint, para definir o Sprint Backlog, que contém todo o trabalho que será executado durante o Sprint.

O Sprint tem duração média de 2 a 4 semanas e são feitas reuniões diárias de acompanhamento (Daily Scrum) do trabalho.

**3.2.1.3 Product Backlog**

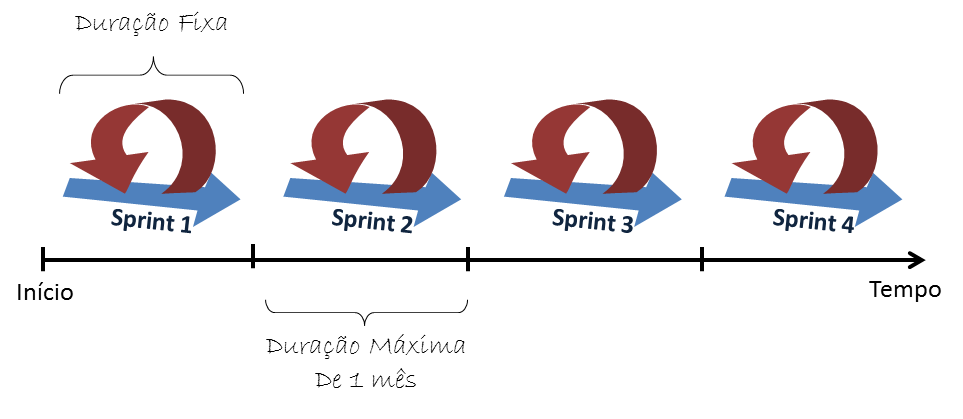
**A close up of a device

Description generated with high confidence**

Figura 5 Product Backlog

*Figura 5 retirada do endereço http://www.mindmaster.com.br/scrum/*

De acordo com Jeff Sutherland, trata-se de uma lista detalhada de tudo que precisa ser feito ou construído para transformar a visão em realidade. Essas Pendências existem e evoluem durante o desenvolvimento do produto; elas são o mapa dele. Em qualquer fase do projeto, são a única e definitiva visão de “tudo que precisa ser feito pela equipe a qualquer momento, em ordem de prioridade”. Só existe uma lista de Pendências; isso significa que o Dono do Produto precisa tomar decisões em relação às prioridade durante todo o processo; ele deve consultar todos os *stakeholders* e a equipe para se certificar de que elas representam tanto o que as pessoas querem, quanto o que pode ser construído.

**3.2.1.4 Sprints**

*Figura 6 retirada do endereço http://www.mindmaster.com.br/scrum/*

Figura 6: Sprints

O produto final é particionado em diversas partes e vai sendo desenvolvido ao longo de cada Sprint. Cada Sprint possui como objetivo entregar uma parte do produto que tenha valor para o cliente. A recomendação é que a duração fixa seja de no máximo um mês.

**3.2.1.5 Sprint Planning**

Esta é a primeira das reuniões Scrum. A equipe, o Mestre Scrum e o Dono do Produto se reúnem para planejar o Sprint. As equipes olham para as tarefas no topo das Pendências e estimam o quanto podem fazer naquele Sprint. Se a equipe já está trabalhando a alguns Sprints, ela deve pegar tarefas que totalizem o mesmo número de pontos do Sprint anterior. Esse número é conhecido como a Velocidade da equipe. O Mestre Scrum e a equipe devem tentar aumentar o número de pontos a cada Sprint.

**3.2.1.6 Daily Scrum ou Stand-Up Meeting**

É uma reunião diária com tempo limite de 15 minutos. Neste tipo de reunião recomenda-se que todos fiquem em pé com a intenção de ser um encontro bem rápido. O Scrum Master faz três perguntas básicas:

1. O que fiz ontem que ajudou o time a atingir a meta do sprint?
2. O que vou fazer hoje para ajudar o time a atingir a meta do sprint?
3. Existe algum impedimento que não permita a mim ou ao time atingir a meta do sprint?

**3.2.1.7 Sprint Review (Revisão do Sprint)**

Qualquer pessoa pode participar, não apenas o Dono do Produto, o Mestre Scrum e a equipe, mas também os *stakeholders*, os gestores, os clientes, e qualquer outra pessoa. Esta é uma reunião aberta na qual a equipe demonstra o que realmente conseguiu concluir com sucesso.

**3.2.1.8 Sprint Retrospective (Retrospectiva do Sprint)**

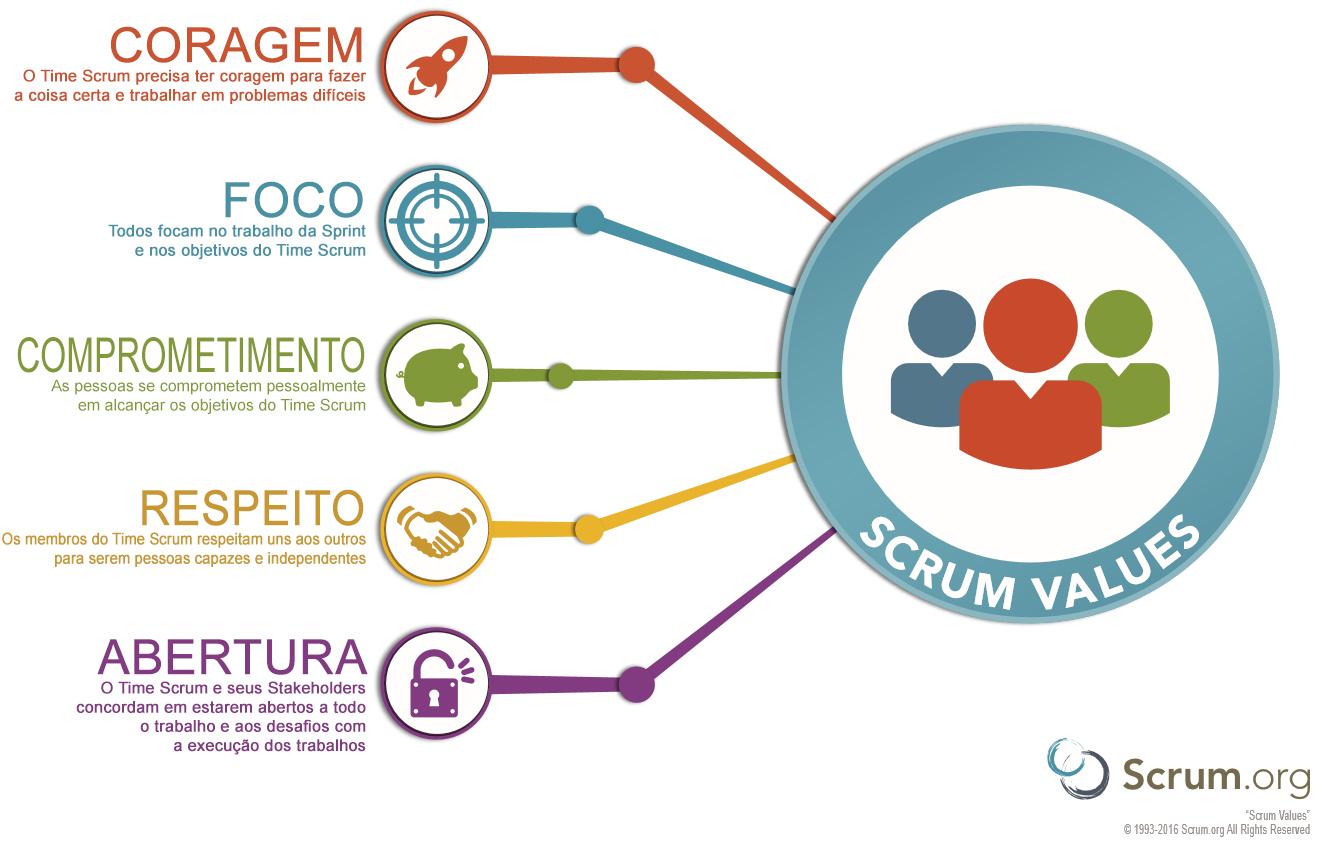
Esta restrospectiva do sprint ocorre depois da Revisão da Sprint e antes da reunião de planejamento da próxima Sprint.

Muitos estudantes, ou até mesmo profissionais, podem confundir a retrospectiva com a revisão. No entanto, enquanto Sprint Review tem como objetivo verificar se o produto necessita de alguma adaptação, o Sprint Retrospective verifica necessidades de adaptações no processo de trabalho.

**3.2 Valores do Scrum**

Quando os valores de comprometimento, coragem, foco, abertura e respeito são assumidos e vividos pelo Time Scrum, os pilares do Scrum de transparência, inspeção e adaptação tornam-se vivos e constroem a confiança para todos. Os membros do Time Scrum aprendem e exploram estes valores à medida que trabalham com os eventos, papéis e artefatos do Scrum (2016, Guia do Scrum 2016, p.5)

Figura 7: Valores do Scrum



Traduzido livremente por Fábio Cruz e localizado no caminho http://www.fabiocruz.com.br/valores-scrum/ .

Original no site da scrum.org no caminho https://www.scrum.org/Resources/What-is-Scrum

**4. Considerações Finais**

a modelagem de dados é algo desafiador para equipes de Tecnologia da Informação (TI), especialmente quando os modelos são desenvolvidos por profissionais diferentes.

# **Referências**

HIROTAKA TAKEUCHI, [IKUJIRO NONAKA](https://hbr.org/search?term=ikujiro+nonaka) - **The New New Product Development Game.** Disponível em https://hbr.org/1986/01/the-new-new-product-development-game. Acesso em 28 de Dezembro de 2017.

DENISSON VIEIRA - **Scrum: A Metodologia Ágil Explicada de forma Definitiva.** Disponível em http://www.mindmaster.com.br/scrum. Acesso em 28 de Dezembro de 2017.

DON WELLS - **Extreme Programming: A gentle introduction**. Disponível em <http://www.extremeprogramming.org/>. Acesso em 29 de Dezembro de 2017.

SUTHERLAND, J. – **Scrum: A arte de fazer o dobro do trabalho na metade do tempo. Leya, 2014.**

ALARIC DAILEY **– Database Design and Naming Conventions** Disponível emhttps://www.codeproject.com/Articles/232457/Database-Design-and-Naming-Conventions. Acesso em 01 novembro de 2017.

DEVMEDIA **– Introdução aos Stored Procedures no SQL Server** Disponível em *[http://www.devmedia.com.br/introducao-aos-stored-procedures-no-sql-server/7904].* Acesso em 22 novembro de 2017.

DATASUS **– MAD Metodologia de Administração de Dados**. Disponível em http://datasus.saude.gov.br/estrutura-mad/norma-mad-menu. Acesso em 31 de outubro de 2017.

DATE, C. J.. **Introdução a Sistemas de Banco de Dados. Elsevier Editora, 2004**.

JEFFREY EDISON **– Naming Coventions in Database Modeling.** Disponível em http://www.vertabelo.com/blog/technical-articles/naming-conventions-in-database-modeling. Acesso em 4 de novembro de 2017.

MSDN **– Especificar Convenções de Nomenclatura (Assistente de Geração de Esquema) (Analysis Services - Dados Multidimensionais)** Disponível em <https://msdn.microsoft.com/pt-br/library/ms177199(v=sql.120).aspx>. Acesso em 01 de outubro de 2017.

SEHROPE SARKUNI **– How I Write SQL, Part 1: Naming Conventions** Disponível em https://launchbylunch.com/posts/2014/Feb/16/sql-naming-conventions/ Acesso em 04 de novembro de 2017.

SILVEIRA, P.; Silveira, G.; LOPES, S.; Moreira, G.; STEPAAT, N. & Kung, F. **–** **Introdução à Arquitetura de Design de Software: Uma Introdução à Plataforma Java** **Elsevier Brasil, 2011**