ISSN 2359-4799
Volume 6 / Número 2 / Ano 2020 – p. 207-227
DOI: 10.36524/ric.v6i2.555

REVISTA Ifes ciência

SUPORTE AO DESENVOLVIMENTO DE *SOFTWARE* BASEADO EM METODOLOGIAS ÁGEIS E FERRAMENTAS CASE EM NUVEM

SUPPORT TO SOFTWARE DEVELOPMENT BASED ON AGILE METHODOLOGIES AND CASE TOOLS IN CLOUD

Artur Schaefer[1]
Julio Cesar Nardi[2*]
[1]Instituto Federal do Espírito Santo - Campus Colatina. E-mail: artur.schaefer2@gmail.com
[2]Instituto Federal do Espírito Santo - Campus Colatina. E-mail: julionardi@ifes.edu.br
*Autor para correspondência

Artigo submetido em 26/03/2020, aceito em 14/07/2020 e publicado em 28/08/2020.

Resumo: Este trabalho apresenta uma pesquisa realizada no contexto de um laboratório de extensão universitária do Instituto Federal do Espírito Santo (Ifes) - Brasil, onde são desenvolvidas aplicações de software para pequenas e médias empresas. As principais dificuldades apresentadas pelo laboratório encontram-se (i) na implantação de novas unidades do laboratório nos vários campi do Ifes; (ii) na condução de projetos de desenvolvimento distribuído de software (entre as suas unidades intercampi); e na condução de projetos dentro de uma mesma unidade de maneira padronizada. Tais dificuldades são consequência da falta de uma padronização metodológica e de um ferramental integrado e de fácil disponibilização/acesso. Assim, esta pesquisa propõe um arcabouço integrado de ferramentas CASE em Nuvem baseado em um processo de software que alinha práticas e metodologias ágeis de desenvolvimento de software, a saber Scrum e XP (*Extremming Programming*). Ademais, este trabalho discute questões relacionadas à adoção de ferramentas CASE considerando a atual tendência de uso dessas ferramentas em Nuvem. Acredita-se que as soluções propostas e discussões apresentadas neste trabalho sejam úteis em outros contextos similares, em que o foco seja a adoção de Computação em Nuvem para suporte ao processo de desenvolvimento de software.

Palavras-chave: Arcabouço integrado; metodologias ágeis; ferramentas CASE em Nuvem.

Abstract: This work presents a research conducted in the context of an university extension laboratory of the Federal Institute of Espírito Santo (Ifes) - Brazil, where software applications have been developed for small and medium-sized enterprises. The main difficulties found in the laboratory concern (i) to the deployment of new units of the lab in the various campi of Ifes; (ii) to the conduction of distributed software development projects (among the existing units); (iii) and even to the conduction of software projects in an unit from a stardardized way. These difficulties are consequence of the lack of a methodological apparatus and of an integrated set of tools that can be easily provided/accessed. Thus, this research proposes an integrated framework of CASE tools in cloud based on a software process that aligns practices and agil metodologies, namely: Scrum and XP (Extremming Programming). Also, this work discusses issues related to adoption of CASE tools considering the current tendency in using such tools in cloud. We believe that the solutions proposed and the discussions addressed in this work can be useful in other similar contexts, where the focus is on adoption Cloud Computing for supporting the software development process.

Keywords: Integrated framework; agile methodology; CASE tools in Cloud.

1 INTRODUÇÃO

Sistemas de software têm exercido um papel essencial na sociedade moderna de maneira que a dependência dos serviços oferecidos por esses sistemas fica evidente a todo instante (ao se efetuar compras pela Internet, comunicação via aplicativos de mensagens, busca de informações sobre determinado local ou produto etc.).

Para aumentar a qualidade desses sistemas e incrementar a produtividade das empresas de software, são utilizados métodos, técnicas e ferramentas, dentre outros, que suportam a construção de software. Assun, ferramentas CASE (*Computer-Aided Software Engineering*) são aplicações que apoiam o trabalho dos desenvolvedores nas várias atividades do processo de construção do software (desde o entendimento do problema, passando pela elaboração da solução e chegando à implantação do sistema no ambiente de produção).

O surgimento das ferramentas CASE e a busca pelo seu alinhamento a um processo de desenvolvimento de software não são recentes. Desde a década de 60 e passando por grande destaque anos 80 (ATKINSON; DRAHEIM, 2013), a adoção das ferramentas CASE vem se intensificando, na medida em que discussões sobre Engenharia de *Software* avançavam motivadas, dentre outros, pela busca por qualidade e produtividade.

A Computação em Nuvem (*Cloud Computing*), por sua vez, começou a tomar proporções significativas a partir de 2007 (MUPPALLA; PRAMOD; SRINIVASA, 2013) e abriu espaço para o surgimento de novos modelos de serviço (*Software-as-a-Service* (SaaS), *Platform-as-a-Service* (PaaS) e *Infraestructure-as-a-Service* (IaaS)) para fornecimento e uso de aplicações de software (BUYYA, 2019).

Deste modo, observou-se uma mudança no perfil das ferramentas CASE utilizadas: passando de ferramentas “locais” (instaladas na máquina do usuário) para ferramentas “em Nuvem” (disponibilizadas pela Internet) (ATKINSON; DRAHEIM, 2013). Com isso, empresas e profissionais de software passaram a usufruir dos benefícios da Computação em Nuvem. O ambiente em Nuvem se coloca, portanto, como um novo paradigma que tem o potencial de gerar valor na construção de aplicações de *software* (KRISHNA; JAYAKRISHMAN, 2013).

Para usufuir, entretanto, do potencial da Engenharia de *Software* baseada em Nuvem (*Cloud-based Software Engineering*) não basta apenas adotar ferramentas CASE em Nuvem e continuar com as mesmas práticas e processos de desenvolvimento de *software* (ATKINSON; DRAHEIM, 2013). É necessário repensar os processos da organização e como um conjunto de ferramentas pode ser alinhado a esses processos de forma a atender às necessidades da organização e às expectativas do cliente (KRISHNA; JAYAKRISHMAN, 2013).

Nesse contexto, estudos recentes (BUTT, 2016; YOUNAS *et al.*, 2018; SEVER, 2019) evidenciam que a combinação de práticas e processos de metodologias ágeis de desenvolvimento com o uso de ferramentas em Nuvem tem apresentado ganhos em qualidade de produto e eficiência de processo de desenvolvimento.

Assim, este trabalho propõe um arcabouço integrado de ferramentas CASE em Nuvem que dê suporte ao processo de desenvolvimento de software. Tal arcabouço é baseado em um processo base cujas atividades e produtos gerados são fundamentados nas metodologias Ágeis Scrum e XP (*Extreme Programming*), usadas de maneira integrada. Dessa maneira, o processo base estabelece requisitos de ferramentas que são, então, atendidos pelo arcabouço proposto. Como resultado, tem-se um conjunto inicial de ferramentas CASE em Nuvem que pode ser atualizado e/ou ampliado quando necessário.

Neste trabalho, utilizou-se como espaço de análise unidades de um

laboratório de Extensão Universitária do Instituto Federal do Espírito Santo (Ifes), cujo foco está no desenvolvimento de aplicações de *software*. Dentre as principais dificuldades enfrentadas no laboratório estão: (i) condução de projetos em uma mesma unidade de maneira padronizada; (ii) condução de projetos de desenvolvimento de software distribuídos (entre as suas unidades *intercampi*); e (iii) implantação de novas unidades nos vários *campi* do Ifes.

O restante deste artigo está estruturado da seguinte maneira: na Seção 2, apresenta-se o método de pesquisa utilizado, caracterizando a pesquisa; na Seção 3, descreve-se o ambiente onde foi identificado o problema de pesquisa; na Seção 4, apresenta-se a fundamentação teórica; na Seção 5, descreve-se o arcabouço de ferramentas CASE em Nuvem proposto, juntamente com o processo base correspondente; na Seção 6, são apresentados e discutidos alguns trabalhos correlatos; e, por fim, na Seção 7 são tecidas algumas considerações finais.

2 MÉTODO E CARACTERIZAÇÃO DA PESQUISA

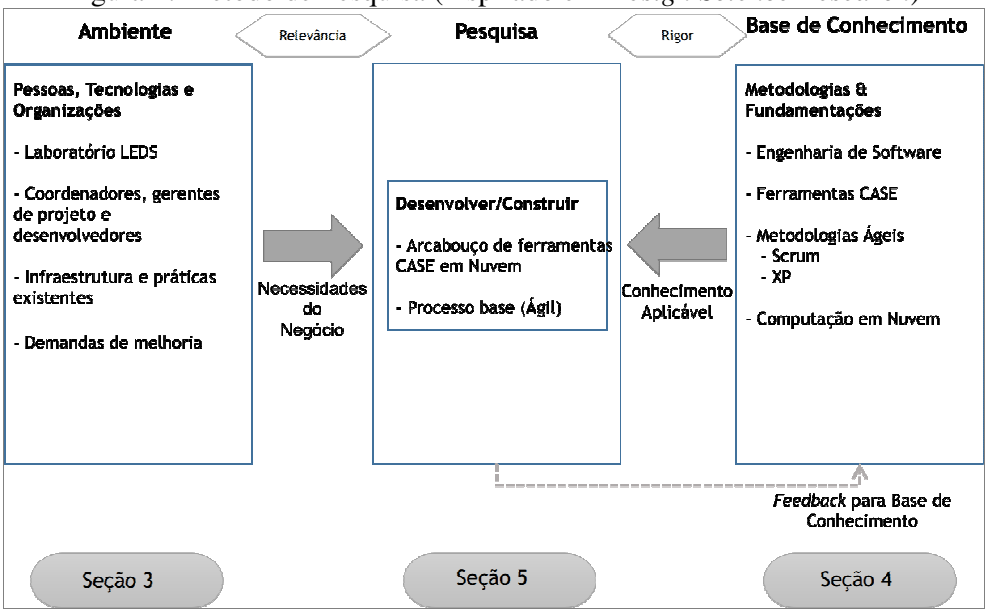
A Figura 1 ilustra os principais elementos abordados nesta pesquisa, organizados segundo o *framework* de *Design Science Research* proposto por Hevner, March e Ram (2004).

Mais à esquerda dessa figura pode-se encontrar os elementos relacionados ao "Ambiente", os quais definem o espaço de problema onde se encontra o fenômeno de interesse. A análise do espaço de problema (formado por pessoas, tecnologias e organizações) permite estabelecer necessidades e problemas que serão objeto de estudo da pesquisa aplicada. No contexto deste trabalho, o ambiente de estudo foram as unidades do Laboratório de Extensão em Desenvolvimento de Soluções (Leds) do Ifes.

Para uma melhor compreensão do ambiente, realizaram-se atividades de coleta e análise de dados utilizando a "técnica de questionários" (MARCONI; LAKATOS, 2003). Utilizaram-se 2 (dois) questionários. Um deles procurou obter informações sobre dificuldades de implantação de uma nova unidade Leds (abordando metodologias, ferramentas padrão, documentações etc.), e de desenvolvimento de projetos distribuídos entre unidades (*intercampi*). Esse questionário foi aplicado ao coordenador geral da Rede LEDS (a qual envolve todas as unidades LEDS). O segundo, mais específico, objetivou obter dados a respeito do uso de ferramentas CASE em Nuvem nas 4 (quatro) unidades LEDS existentes. O público-alvo desse questionário foram os coordenadores de cada unidade LEDS. O motivo da escolha dos participantes dos questionários é justificado pela visão geral de todo o processo de desenvolvimento que eles possuem. A apresentação dos dados coletados, bem como a caracterização dos problemas e necessidades identificadas, são detalhadas na Seção 3.

Mais à direita da Figura 1, podem-se observar os elementos relacionados à "Base de Conhecimento", a qual provê o "estado da prática" e o "estado da arte" sobre o tema pesquisado e, consequentemente, fornece o aparato teórico e metodológico para o desenvolvimento da pesquisa. Tais elementos estão descritos na Seção 4.

Figura 1: Método de Pesquisa (inspirado em *Design Science Research)*



Fonte: Autores (inspirados em: Hevner, March e Ram, 2004)

Ao centro da Figura 1, encontram-se os elementos relacionados ao processo "Desenvolver/Construir", o qual envolve, basicamente, o *design* e a construção e da solução proposta. Assim, destacam-se os seguintes passos: (i) estudo das abordagens de Desenvolvimento Ágil de *software* com foco no Scrum e XP; (ii) construção de um alinhamento entre as etapas e entregas do Scrum e com as práticas do XP; (iii) definição de processo de desenvolvimento base fundamentado no alinhamento realizado entre Scrum e XP; (iv) estudo e levantamento de potenciais ferramentas CASE em Nuvem a serem utilizadas; e (v) definição do conjunto de ferramentas a ser utilizado no arcabouço proposto e alocação dessas ferramentas ao processo base definido. Os elementos produzidos ao longo da construção do arcabouço proposto constam na Seção 5.

3 LABORATÓRIO DE EXTENSÃO EM DESENVOLVIMENTO DE SOLUÇÕES

O LEDSLeds do Ifes é um espaço de desenvolvimento de soluções de *software* onde alunos dos cursos de computação podem, por meio da participação em projetos reais, alinhar teoria e prática com foco na resolução de problemas demandados pela sociedade.

Atualmente, existem quatro unidades Leds sediadas nos seguintes *campi*: Colatina, Guarapari, Santa Teresa e Serra. Tais unidades formam o Programa Institucional de Extensão em Rede chamado "Rede Leds". Entretanto, devido às particularidades de cada *campus*, maturidade da unidade e demandas regionais, cada unidade Leds acabou por criar sua própria infraestrutura de ferramentas de *software* e definir metodologias próprias para dar suporte ao desenvolvimento dos respectivos projetos.

Este trabalho levantou informações da Rede Leds bem como de suas unidades, por meio de questionários aplicados ao coordenador geral da rede e aos coordenadores de cada unidade. A partir das informações obtidas, identificaram-se necessidades, potencialidades e requisitos relacionados à adoção de ferramentas CASE, os quais são apresentados a seguir.

Necessidades:
A falta de um arcabouço de ferramentas CASE padrão que possa ser compartilhado dificulta (i) a definição de uma prática padrão dentro de uma mesma unidade Leds,

(ii) o desenvolvimento (em conjunto) de projetos distribuídos e (iii) a implantação de novas unidades Leds.

Potencialidades:

- As 2 (duas) unidades Leds com mais tempo de atuação (Leds-Colatina e Leds-Serra), dentre as 4 (quatro) unidades existentes, já possuem experiência com o uso de ferramentas CASE em Nuvem. Isso parece indicar que uma solução baseada em ferramentas CASE em Nuvem possui um contexto de receptividade favorável.
- Os coordenadores da Rede Leds e das suas unidades reconhecem vantagens de se adotar ferramentas CASE em Nuvem, dentre elas: (i) menor preocupação com a manutenção da infraestrutura física de servidores e com instalação e configuração de *softwares* nas máquinas locais (minimizando conflitos de versões); (i) *backup* dos arquivos fica na nuvem; e (iii) facilidade de acesso aos arquivos dos projetos de qualquer local com acesso à Internet.
- Os aspectos metodológicos e diretrizes de desenvolvimento de *software* aplicados nas unidades Leds são fortemente baseados nos princípios e valores das Metodologias Ágeis de Desenvolvimento de *Software* (em especial Scrum e *Extremming Programming* (XP)).

Requisitos da solução:

- RQ1 - O arcabouço a ser definido deve ser baseado em propósitos e tipos de ferramenta e não em ferramentas específicas. Assim, é possível substituir, caso necessário, ferramentas do arcabouço por outras mais apropriadas.
- RQ2 - Deve ser possível utilizar também ferramentas CASE "locais", pois pode não haver, para um propósito específico, uma ferramenta CASE em Nuvem como alternativa viável.
- RQ3 - As ferramentas devem prover mecanismos de comunicação, entre elas, que permitam sua integração.
- RQ4 - Deve ser desenvolvida uma base metodológica e práticas padrão baseadas nas metodologias ágeis Scrum e XP de forma a guiar a utilização dessas ferramentas CASE.

As necessidades, potencialidades e requisitos de solução foram utilizados como base para motivação, desenvolvimento e avaliação da solução proposta.

4 FUNDAMENTAÇÃO TEÓRICA

Nesta seção, são apresentados os principais conceitos que compõem a base de conhecimento/teórica deste trabalho.

4.1 FERRAMENTAS CASE E COMPUTAÇÃO EM NUVEM

O desenvolvimento de *software* com qualidade, produtividade, obedecendo a prazos estabelecidos e sem a necessidade de alocação de mais recursos, tem sido um desafio para a Engenharia de *Software* (SOMMERVILLE, 2007). Nesse contexto, o papel das ferramentas CASE (*Computer-Aided Software Engineering*) é fundamental (ATKINSON; DRAHEIM, 2013).

A gama de ferramentas CASE inclui editores de código, dicionários de dados, compiladores, depuradores, gerenciadores de versão etc. (SOMMERVILLE, 2007). De maneira geral, toda ferramenta que auxilie o desenvolvimento de tarefas ao longo do processo de desenvolvimento de *software* pode ser considerada uma ferramenta CASE.

Ainda que muitas dessas ferramentas forneçam um suporte pontual à determinada tarefa (edição de texto ou depurador, por exemplo), é importante sempre buscar a integração entre essas ferramentas, de maneira que os dados capturados e manipulados por uma ferramenta possam servir de entrada/apoio na operação de outras ferramentas. Com isso, potencializa-

se a automatização da execução do processo de desenvolvimento de *software*.

Com a ampliação do uso da Internet, melhoria das conexões de rede e desenvolvimento de métodos computacionais mais eficientes, ampliaram-se o uso e o interesse em tecnologias baseadas em Internet, como é o caso da Computação em Nuvem (*Cloud Computing*) (SAHMIM; GHARSELLAOUI, 2017).

A Computação em Nuvem não se refere a uma tecnologia específica, mas um modelo de provisão de recursos/capacidades que envolvem infraestrutura de tecnologia da informação, componentes e aplicações e que tem sido cada vez mais integrado às operações de negócio das organizações (BENLIAN *et al.*, 2018; SUNYAEV, 2020).

Algumas das vantagens da adoção da Computação em Nuvem são (TRITSINIOTIS, 2013; BUTT, 2016):

Economia em investimentos de *hardware* e *software*: diminui o custo com aquisição de máquinas e/ou novas versões de *softwares*, já que essa responsabilidade passa a ser da empresa provedora da aplicação em Nuvem.

Trabalho de qualquer lugar. Acesso às aplicações/recursos na Nuvem pode ser feito de qualquer local que tenha acesso à Internet.

Preço por demanda de uso. Contratos de serviços em Nuvem, normalmente, seguem o conceito de "*pay-as-you-go*" em que somente é cobrado pelo tempo/recurso que utilizar.

Escalabilidade: conforme aumenta a necessidade de mais recursos (armazenamento, processamento etc.), modelos de provimento de serviços em Nuvem permitem lidar com esse crescimento por demanda.

***Softwares* e tecnologias sempre atualizadas**. A atualização dos servidores e *softwares* é feita pela empresa prestadora do serviço em Nuvem e evita que o contratante se preocupe com tais questões.

Segurança. As informações são mantidas na Nuvem e garantidas pelo prestador de serviços. A perda de um *notebook* ou computador local não afeta os dados.

Competitividade. A empresa pode se manter focada no que importa. A Computação em Nuvem ajuda a empresa a delegar as questões de infraestrutura de *hardware* e *software*.

Assim, como vários outros tipos de aplicações de *software*, as ferramentas CASE passaram a ser providas e utilizadas em Nuvem impactando a maneira com que as empresas de desenvolvimento de *software* conduzem seus processos. Isso tem levado ao que Atkinson e Draheim (2013) chamam de Abordagem Orientada a Nuvem para Engenharia de *Software* (*cloud-driven approach to software engineering*) e que levou esses pesquisadores a cunharem o termo CASE 2.0 (*Cloud-Aided Software Engineering* - CASE 2.0), cujos fundamentos se estendem desde as ferramentas CASE até os modelos de ciclo de vida e demais práticas de construção de *software*. Nessa linha, a simples adoção de ferramentas CASE em Nuvem mantendo tradicionais práticas de desenvolvimento limita o potencial transformador de abordagens de Engenharia de *Software* em Nuvem (ATKINSON; DRAHEIM, 2013). É necessário, portanto, pensar em metodologias práticas e mais dinâmicas.

4.2 METODOLOGIAS ÁGEIS DE DESENVOLVIMENTO DE *SOFTWARE*: SCRUM e XP

Metodologias ágeis de desenvolvimento de *software* são uma alternativa às ditas "metodologias tradicionais" de desenvolvimento de *software* (SOARES, 2004) e buscam tornar tal atividade mais dinâmica.

Há várias metodologias classificadas como ágeis, dentre elas, destacam-se o Scrum e o *Extremming Programming* (XP) (ASHRAF; AFTAB, 2017).

O Scrum, a mais popular das metodologias ágeis (ASHRAF; AFTAB,

2017), define um processo de desenvolvimento iterativo, sendo útil, principalmente, quando os requisitos não estão bem definidos ou há alta frequência de mudanças (SOARES, 2004). Tal metodologia define papéis desempenhados pelos agentes envolvidos no processo de desenvolvimento, sendo eles: o *Product Owner* (quem define os requisitos do projeto); *Scrum Master* (gerente do projeto, quem distribui as tarefas e gerencia a equipe); e o *Development Team* (quem é responsável por executar as tarefas de codificação e teste necessárias) (SILVA; SOUZA; CAMARGO, 2013).

Os principais eventos e artefatos que fazem parte do processo do Scrum são descritos a seguir (SABBAGH, 2014; SCHWABER, SUTHERLAND, 2017):

Artefatos

Product Backlog: lista dos requisitos (características e funcionalidades) desejadas pelo *Product Owner* para um produto. Essa lista não precisa estar completa desde o início do projeto e pode evoluir conforme andamento o desenvolvimento do *software*.

Sprint Backlog: lista de requisitos a serem contemplados em uma *Sprint*, contendo quem será o responsável pela implementação de cada requisito.

Increment: também chamado de "incremento de *software*" ou simplesmente "produto". É o resultado (ou entrega) do que foi feito durante uma *Sprint*. O resultado/entrega pode ser uma versão intermediária do *software* ou o produto final.

Eventos

Sprint: representa um ciclo de trabalho (iteração) do Scrum. As *Sprints* possuem duração de uma a quatro semanas e apresentam como meta implementar o que foi definido no *Sprint Backlog*.

Sprint Planning: reunião de planejamento entre os envolvidos no projeto a fim de definir quais requisitos são prioritários e serão incluídos no *Sprint Backlog* da próxima *Sprint*.

Daily Scrum: reunião diária, curta e direta entre *Scrum Master* e *Team Development*. São tratados assuntos do dia para alinhar a equipe. Em geral, são realizadas pela manhã para saber o que foi feito no dia anterior e o que será feito durante o dia.

Sprint Review: reunião de, no máximo, quatro horas para discutir e revisar o que foi feito durante uma *Sprint* como, por exemplo, avaliar o incremento de *software* gerado, receber *feedback* do *Product Owner*, revisar orçamento e cronograma.

Sprint Retrospective: realizada após o *Sprint Review*. Reunião de, no máximo, três horas. Possui foco no *Development Team*. É uma oportunidade de avaliar as práticas e tarefas que ocorreram na *Sprint*, sendo essencial a participação do *Scrum Master*. As pessoas envolvidas são pautas de discussão nessa etapa, pois é necessário ter uma equipe motivada e produtiva.

O *Extremming Programming* (XP) define práticas de engenharia a serem aplicadas pela equipe durante o desenvolvimento do *software*. As práticas propostas pelo XP são (BECK, 2004):

***Test-Driven-Development* (TDD)**: estratégia de desenvolvimento que define que os testes de *software* devem ser escritos antes mesmo do código-fonte. Assim, os testes podem ser automatizados e executados em tempo de desenvolvimento.

The Planning Game: dinâmica de estimativa e planejamento envolvendo desenvolvedores e cliente a fim de, dentre outros, priorizar e estimar tempo e esforço para os requisitos a serem desenvolvidos.

On-Site Customer: o cliente se faz presente ao longo do processo de desenvolvimento, tendo papel importante na definição/escrita dos testes, definição do escopo, ajustes no projeto, priorização das demandas, esclarecimento de requisitos e validação de resultados/entregas.

Pair Programming: consiste em ter o código gerado por duas pessoas simultaneamente: enquanto uma escreve, a outra revisa. De tempos em tempos, as funções podem ser alternadas.

Code Refactoring: visa ao melhoramento contínuo do código buscando remoção de redundâncias, de funções desnecessárias, simplificação de *design* etc.

Continuous Integration: a integração contínua define que os módulos e trechos de códigos devem ser continuamente integrados e que passem por testes antes e após as mudanças.

Small releases: consiste em definir e efetuar entregas do *software* com um escopo reduzido, mas frequentes. Assim, tem-se um *feedback* rápido e frequente do cliente/usuário.

Simple Design: consiste em criar um código simples e funcional com o que foi definido no escopo do projeto. Códigos complexos, com pouca modularização e extensos, geram confusão ao serem revistos e complicam tarefas de correções/melhorias.

Collective Code Ownership: todo o código gerado deve estar acessível e conhecido por toda a equipe de desenvolvimento e demais envolvidos.

Coding Standards: o uso de padrões de codificação, assim como da prática *Collective Code Ownership*, facilita o trabalho em equipe e o desenvolvimento de um *software* de maior qualidade.

System Metaphor: a equipe deve seguir uma linguagem comum para que se evite discordâncias terminológicas. As metáforas são definidas para evitar confusão de termos durante o desenvolvimento do projeto.

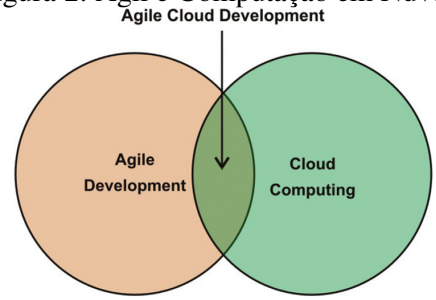
40-Hour Week: toda a equipe envolvida no projeto deve estar imersa em um ambiente de trabalho agradável e que não exija esforço demasiado, mantendo um ritmo e um planejamento sustentável de trabalho.

Enquanto o Scrum é focado em práticas de gerenciamento do projeto e da organização, o XP tem foco nas tarefas relacionadas ao desenvolvimento técnico do *software*. As duas abordagens apresentam focos distintos, mas complementares sendo, dentre as metodologias ágeis, aquelas comumente utilizadas de maneira integrada (ASHRAF; AFTAB, 2017).

4.3 METODOLOGIAS ÁGEIS E COMPUTAÇÃO EM NUVEM

O alinhamento entre Metodologias Ágeis e Computação em Nuvem tem sido reconhecido como uma boa opção na busca por melhoria no desenvolvimento de *software* (KARUNAKARAN, 2013). Tal alinhamento pode promover melhoria na eficiência do processo, no desempenho dos desenvolvedores e na qualidade do produto gerado (BUTT, 2016). Nesse sentido, cada vez mais estudos têm sido conduzidos, a fim de entender e explorar os pontos de alinhamento/intersecção entre essas duas áreas, cf. Figure 2 (BUTT, 2016) (YOUNAS *et al.*, 2017).

Figura 2: Ágil e Computação em Nuvem



Fonte: Butt (2016) e Younas *et al.* (2017).

Tais estudos têm evidenciado que o desenvolvimento ágil requer *feedback* rápido entre *stakeholders* ao longo do processo, o que a Computação em Nuvem pode promover (KARUNAKARAN, 2013). Além disso, algumas outras vantagens desse alinhamento são: (i) desenvolvimento distribuído de *software*, (ii) compartilhamento de dados, (iii) priorização de tarefas; (iv) transparência nas ações realizadas, (v) infraestrutura básica imediata, e (vi) benefícios para os usuários de negócio.

Por fim, o desenvolvimento ágil é fundamentado no forte envolvimento da equipe e dos clientes de negócio no âmbito de ciclos iterativos, nos quais o trabalho colaborativo se baseia em intensa comunicação. Nesse sentido, academia e indústria reconhecem que a Computação em Nuvem pode favorecer não apenas a comunicação no nível técnico, mas também no nível de negócio, promovendo melhoria e mais rapidez na troca de informações (KARUNAKARAN, 2013).

5 UM ARCABOUÇO INTEGRADO DE FERRAMENTAS CASE EM NUVEM

O arcabouço proposto incorpora práticas de Engenharia de *Software* que visam a apoiar o aumento de produtividade e de qualidade do trabalho de desenvolvedores de *software* por meio de um conjunto integrado de ferramentas CASE em Nuvem, cujo uso é orientado por um processo de *software* baseado em práticas de metodologias ágeis: Scrum e XP.

5.1 DEFINIÇÃO DE UM ALINHAMENTO ENTRE XP E SCRUM

Neste trabalho, buscou-se definir um alinhamento entre as práticas do XP e os eventos do Scrum. O propósito desse alinhamento é a definição de um processo base de desenvolvimento de *software* fundamentado em metodologias ágeis e que seja suportado por ferramentas CASE em Nuvem. O Quadro 1 apresenta uma visão relacional desse alinhamento e explicita os aspectos de processo a serem suportados.

Como se pode notar, as práticas do XP de caráter mais geral tendem a estabelecer uma relação transversal, passando por vários eventos/etapas do Scrum, como é o caso das práticas *The Planning Game*, *System Metaphor*, *40-Hour Week*. Essas práticas são exploradas tanto ao longo de eventos de planejamento (*Sprint Planning*) e construção (*Sprint*) quanto em eventos de avaliação (*Daily Scrum*, *Sprint Review*, *Sprint Retrospective*).

Com foco no planejamento, as práticas *The Planning Game* e *Small Releases* se relacionam com eventos Scrum que buscam planejar e (re)avaliar o andamento do projeto e o produto (*Sprint Planning*, *Daily Scrum*, *Sprint Review* e *Sprint Retrospective*).

Práticas com foco mais técnico tendem a se relacionar à construção do *software* em si. Assim, pode-se observar que a adoção de *Test Driven Development*, *Pair Programming*, *Code Refactoring*, *Continuous Integration*, *Simple Design*, e *Coding Standards* do XP alinham-se mais com a execução do evento *Sprint* do Scrum.

Durante o *Sprint Planning,* as práticas *The Planning Game* e *Small Releases* desempenham um papel importante, na medida em que apoiam a definição de questões mais objetivas, relacionadas ao planejamento de uma *Sprint*, como, por exemplo, priorização de requisitos e estimativas, culminando, assim, na definição do *Sprint Backlog*. Nessa etapa, a presença do cliente é essencial (*On-site Customer*), pois ele, dentre outros, tem a função de priorizar os requisitos a serem implementados na próxima *Sprint*. O planejamento da *Sprint* precisa ainda estar baseado em um desenvolvimento sustentável do produto, sob a perspectiva do esforço semanal exigido da equipe de desenvolvimento (*40-Hour Week*) alinhado à busca por entregas frequentes e *feedback* rápido do cliente (*Small Releases*). Por fim, por meio da definição de uma linguagem comum (*System Metaphor*) (vocabulário, metáforas etc.) que apoiem a comunicação entre os envolvidos nas atividades de planejamento, busca-se minimizar falhas de comunicação e ampliar o entendimento do que o produto final deve atender.

Ao longo da *Sprint*, mesmo se tratando de uma etapa mais técnica de construção de código, a presença e a participação do cliente (*On-site Customer*) é importante, pois permite que esse dê o *feedback* necessário e rápido a respeito das funcionalidades construídas e esclareça requisitos que, eventualmente, não estejam

bem definidos. Nesse contexto, a utilização de um vocabulário comum para minimizar problemas de comunicação (*System Metaphor*) também se faz muito importante.

Durante o desenvolvimento da *Sprint*, a manutenção do ritmo de trabalho equilibrado deve ser buscado e tratado em conformidade com o planejamento. As demais práticas aplicadas, no âmbito da *Sprint,* visam a suportar a construção propriamente dita do código-fonte e dos aspectos de *design* do produto a ser desenvolvido.

Quadro 1: Relação entre as Práticas XP & Eventos Scrum e aspectos a serem suportados

XP	***Scrum***					**Aspectos a serem suportados por ferramentas CASE**
	Sprint Planning	***Sprint***	***Daily Scrum***	***Sprint Review***	***Sprint Retros-pective***	
On-site Customer	X	X		X	X	- Cliente participativo, que seja ciente e corresponsável pelas atividades.
System Metaphor	X	X	X	X	X	- Definição e uso de linguagem/vocabulário compartilhado.
40-Hour Week	X	X	X	X	X	- Promoção do desenvolvimento sustentável.
The Planning Game	X	X	X	X	X	- Definição, monitoramento e avaliação do planejamento.
Small Releases	X					- Versões de escopo reduzido disponíveis para *feedback* rápido ao fim da *Sprint*.
Test Driven Development		X				- Definição, implementação e execução automatizada de testes prévios ao desenvolvimento do código-fonte. - Registro de erros e não-conformidades. - Acompanhamento de solução/ajustes.
Pair Programming		X				- (Re)construção de testes e código-fonte em dupla. - Análise e melhoria da qualidade do código-fonte e do *software* produzido.
Code Refactoring		X				
Coding Standards		X				
Simple Design		X				
Continuous Integration		X				- Automatização do processo de *build* integrada a outras tarefas/processos (ex: controle de versão e publicação).
Collective Code Ownership		X				- Acesso compartilhado ao código. - Responsabilidade compartilhada.

Fonte: elaborado pelos autores

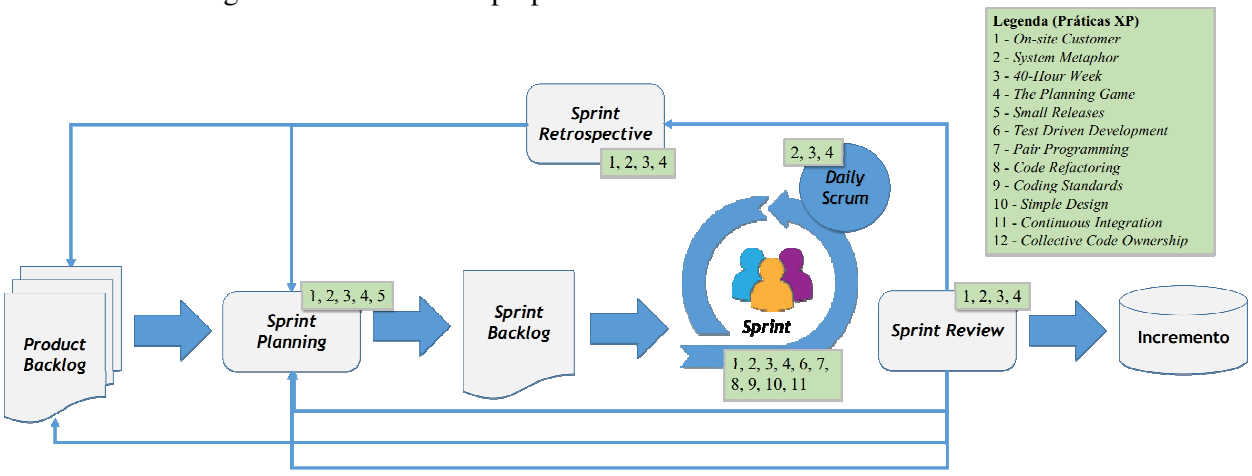
Os eventos do Scrum, voltados para atividades de avaliação e *feedback* (*Sprint Review* e *Sprint Retrospective*), estão diretamente relacionados à presença e à participação do cliente (*On-site Customer*). Nesses eventos, utiliza-se uma linguagem comum (*System Metaphor*) e se realizam avaliações do planejamento e dos procedimentos adotados, a fim de se obter

ajustes, caso necessário (*40-Hour Week, Small Releases* e *The Planning Game*).

A Figura 3 apresenta outra visão do alinhamento proposto entre XP e Scrum, explorando o aspecto processual. Buscou-se destacar que o sequenciamento dos eventos, os artefatos produzidos e os consumidos são baseados no Scrum, enquanto aspectos mais voltados para a construção do *software* são aplicados ao longo desse processo e obtidos do XP.

A partir do alinhamento estabelecido e da definição do processo base, foi possível identificar aspectos que devem ser suportados pelo ferramental integrado a ser utilizado. Assim, ao longo do processo há ferramentas CASE que auxiliam a equipe envolvida (desenvolvedores, gerentes e cliente/usuários). Tais ferramentas precisam estar disponíveis e se comunicar, a fim de possibilitar a integração de informação e funcionalidades. Essa integração permitirá um maior suporte às atividades de desenvolvimento de *software*. Na próxima seção, é apresentado um conjunto possível de ferramentas para suportar o processo base definido, como se comunicam e quais as dependências entre elas.

Figura 3: Processo base proposto: Práticas do XP alinhadas ao Scrum



Fonte: Autores - baseados em (CASAGRANDE, 2018)

5.2 UM CONJUNTO DE FERRAMENTAS PARA O ARCABOUÇO

Além dos aspectos descritos na seção anterior, a serem suportados pelas ferramentas, o potencial de integração, a disponibilidade na Nuvem e a licença de uso (preferência por ferramentas gratuitas e de código aberto) foram levados em consideração para a seleção do conjunto de ferramentas. As ferramentas selecionadas são apresentadas a seguir.

Slack: Canal centralizado de comunicação a ser utilizado por toda a equipe envolvida no projeto (cliente, usuários e desenvolvedores). Permite a integração com outras ferramentas, a criação de canais de comunicação e o envio de mensagens. Pode ser utilizada de maneira a comunicar eventos e tarefas realizadas no âmbito de outras ferramentas integradas a ela como, por exemplo, atualizações de código-fonte, de planejamento e execuções de testes. Permite a comunicação direta entre os membros da equipe, registrando o histórico de conversas, servindo como evidências documentais.

Taiga: auxilia o gerenciamento de projetos. Possui versões pagas e a gratuita. Na versão gratuita, é possível publicar um projeto privado e inúmeros projetos públicos. É possível definir um quadro de atividades para realizar a gerência do projeto. As mudanças realizadas no projeto são notificadas aos envolvidos por meio de canais de comunicação definidos no *Slack*,

como forma de manter a equipe informada sobre o andamento do projeto.

Bitbucket: baseado na ferramenta *Git*, armazena e realiza gerenciamento de versão do código fonte e código de teste, bem como de outros artefatos gerados ao longo dos projetos. Possui planos de uso gratuitos e pagos. Fornece API (*Application Integration Interface*) para consulta e alterações no repositório. Permite criar *wiki* para o projeto, onde se pode manter documentos ou descrições compartilhadas sobre o domínio de aplicação para o qual o *software* está sendo construído. Para o ferramental proposto, o *Slack* será acionado a partir de qualquer alteração no repositório.

Travis CI: dá suporte à integração contínua. É uma ferramenta gratuita para repositórios públicos no *Bitbucket*. Para repositórios privados, existem versões pagas, cujo custo varia, conforme demanda da equipe. Dá suporte a *builds* automatizados com verificação de erros. Além da integração com o *Bitbucket*, permite a integração com o *Slack*. Dessa forma, quando houver atualização do *build* do projeto (com êxito ou falha), ele notifica os canais de comunicação definidos no *Slack*.

Sonarcloud: Ferramenta *web* para verificação de qualidade de código, detecção de *bugs* e vulnerabilidades. É construído a partir do *SonarQube* e possui integração com o *Travis CI*. Ao ser ativado pelo *Travis C*I, apresenta um painel com métricas/indicadores de qualidade de código.

Além das ferramentas supracitadas disponíveis na Nuvem, utilizaram-se duas outras que rodam localmente na máquina do desenvolvedor. Elas estão disponíveis nas plataformas Linux e Windows. Tais ferramentas podem ser integradas àquelas na Nuvem.

***Docker*:** é um Linux *Container* (LXC) utilizado para virtualização de sistemas, sendo uma ferramenta *open source* que permite que aplicações de *software* executem em *containers*, facilitando a portabilidade de aplicação, o isolamento dos processos executados na máquina, a prevenção de violação externa na aplicação e o gerenciamento de consumo de recursos. O principal fator para escolha do *Docker* é a padronização da infraestrutura de desenvolvimento. A partir de sua configuração, a aplicação está pronta para execução e a infraestrutura pode ser replicada para desenvolvedores no projeto.

IDE Eclipse: ferramenta usada para construção de código-fonte. Embora haja várias ferramentas de edição de código-fonte disponíveis, decidiu-se pelo *Eclipse* pela facilidade de uso, pelas funcionalidades que podem ser incorporadas e por possuir uma comunidade grande e ativa.

O Quadro 2 apresenta a relação estabelecida entre as ferramentas apresentadas e as práticas do XP e os eventos do Scrum.

Primeiramente, no que tange ao Scrum, uma descrição do uso de cada ferramenta é apresentada abaixo.

IDE Eclipse: utilizado no desenvolvimento do produto (código-fonte) e da pirâmide de testes. O uso dessa ferramenta se dá, fundamentalmente, durante a *Sprint*.

Docker: assim como o *IDE Eclipse*, o *Docker* será utilizado na *Sprint* para auxiliar os desenvolvedores durante o processo de criação e controle do *software*. Tal ferramenta conterá a infraestrutura /ferramental básico de desenvolvimento como, por exemplo, sistema gerenciador de banco de dados, bibliotecas necessárias etc. utilizadas ao longo do desenvolvimento da *Sprint*. Todas as configurações de infraestrutura se concentram em *containers* específicos e, caso seja necessário, importa-se o *container* desejado e mantém-se a mesma configuração para todos os desenvolvedores.

Quadro 2: Relação entre ferramentas selecionadas e Práticas XP & Eventos Scrum

XP	***Scrum***					**Aspectos a serem suportados por ferramentas CASE**
	Sprint Planning	***Sprint***	***Daily Scrum***	***Sprint Review***	***Sprint Retros-pective***	
On-site Customer	*Taiga Slack*	*Taiga Slack*		*Taiga Slack*	*Taiga Slack*	- Cliente participativo, ciente e corresponsável pelas atividades.
System Metaphor		*Bitbucket*				- Definição e uso de linguagem/vocabulário compartilhado.
40-Hour Week						- Promoção do desenvolvimento sustentável.
The Planning Game	*Taiga Slack*		*Taiga Slack*	*Taiga Slack*	*Taiga Slack*	- Definição, monitoramento e avaliação do planejamento.
Test Driven Development		*Eclipse Docker* / *Taiga Travis CI Sonar-cloud*				- Definição, implementação e execução automatizada de testes prévios ao desenvolvimento do código-fonte. - Registro de erros e não-conformidades. - Acompanhamento de solução/ajustes.
Pair Programming		*Taiga*				- (Re)construção de testes e código-fonte em dupla. - Análise e melhoria da qualidade do código-fonte e *software* produzido.
Code Refactoring		*Sonar-cloud*				
Coding Standards		*Sonarcloud*				
Simple Design		*Sonarcloud*				
Continuous Integration		*Travis CI Slack Bitbucket*				- Automatização do processo de *build* integrada a outras tarefas/processos (ex: controle de versão e publicação).
Small Releases		*Travis CI*				- Versões de escopo reduzido disponíveis para *feedback* ao fim da *Sprint*.
Collective Code Ownership		*Bitbucket Travis CI Slack*				- Acesso compartilhado ao código. - Responsabilidade compartilhada.

Fonte: Elaborado pelos autores.

Bitbucket: armazena, na Nuvem, os códigos gerados pela equipe de desenvolvedores durante a *Sprint*, contendo as *releases* do produto gerado. Pode ainda ser utilizado durante reuniões tais como o *Daily Meeting*, *Sprint Review* e *Sprint Retrospective*, pois fornece informações sobre o processo de alterações do *software*.

Taiga: presente em todas as etapas da metodologia Scrum, desde a criação do *Product Backlog*, do *Sprint Backlog*, do avanço da *Sprint*, da análise de desempenho e planejamento da equipe etc. Assim, apoia a gerência da equipe e do projeto.

Slack: está presente em todas as etapas do Scrum, pelo fato de ser o centro de comunicação do arcabouço proposto. Todas

as notificações sobre mudanças e avanços do projeto passam e são redirecionadas por meio do *Slack*.

Travis CI: dá suporte à integração contínua durante as atividades de construção do código-fonte do *software* e dos casos de teste ao longo da *Sprint*. Assim, além da automatização na geração dos *builds* e verificação de erros, aciona outras ferramentas, como *Bitbucket* e *Slack*. É uma ferramenta de infraestrutura do processo de desenvolvimento, visando a integrar outras ferramentas.

Sonarcloud: ao longo da *Sprint*, dá suporte à análise da qualidade e funcionalidade dos códigos gerados (tanto do *software* quanto dos casos de teste). Pode ser utilizada tanto por desenvolvedores individualmente ou em reuniões de equipe.

Como supracitado, tais ferramentas foram pensadas de modo a também dar suporte às práticas do XP. As relações entre práticas XP e o conjunto definido de ferramentas são descritas abaixo.

Test Driven Development (TDD): para apoiar o TDD, há diversas ferramentas e suítes disponíveis como, por exemplo, *JUnit*, *Jasmine*, *PyUnit* e *TesteNG*. Tais suítes estão relacionadas, em geral, à linguagem de programação adotada no desenvolvimento do *software*. Sendo assim, não se definiu uma ferramenta específica, pois se entende que em cada projeto a equipe pode definir a ferramenta mais adequada. Como infraestrutura comum, entretanto, pode-se citar o suporte que o *Travis CI* oferece ao acionamento dos casos de teste automatizados e à análise da qualidade do código gerado. Ademais, a ferramenta *Taiga* pode ser utilizada para efeito de planejamento e monitoramento das atividades de teste.

The Planning Game: o planejamento poderá ser realizado por meio do *Taiga*, colocando os *user stories*, os requisitos do projeto, o andamento das *Sprints* etc.

On-site Customer: os clientes, de uma maneira geral, não necessitam de acesso direto às ferramentas de construção de código e/ou ferramentas de infraestrutura de desenvolvimento (p.ex., integração contínua - *Travis CI*). Entretanto, de maneira indireta, muitas das informações geradas por essas ferramentas poderão ser oferecidas a eles. Outras ferramentas, por sua vez, voltadas mais para o acompanhamento do projeto (p.ex., *Taiga* e *Slack*) poderão ser acessadas diretamente pelos clientes como forma de estreitar a comunicação e integração com toda a equipe de projeto.

Pair Programming, Coding Standards e ***Code Refactoring***: essas práticas estão atreladas ao desenvolvimento do código-fonte, em si, e podem ser suportadas de várias maneiras e em diversos níveis de automatização. No caso do conjunto de ferramentas proposto, destacam-se o *IDE Eclipse* para suporte a *Code Refactoring* e o Sonarcloud para verificação de *Coding Standards*.

Continuous Integration: a integração contínua é a principal função do *Travis CI*. A cada mudança do *software* enviada para o repositório na Nuvem (no *Bitbucket*) dispara a realização de testes e a geração de um novo *build*.

Small Releases: as pequenas versões de *software* são disponibilizadas via *Bitbucket*. O controle de versões proporcionado pelo *Bitbucket* permite acesso e identificação única das versões do produto;

Simple Design: por meio do *Sonarcloud*, tem-se o apoio à análise do código em busca de possíveis pontos de melhoria e não aderências ao padrão de condificação (*Code Standard*). Assim, essa ferramenta auxilia na busca por códigos que possam ser refatorados e simplificados, visando a melhorias no *software* e atendendo diretamente a essa prática.

Collective Code Ownership: o *Slack* servirá como uma central de notificações para atualizações e mudanças nos arquivos do projeto, suportando essa prática de propriedade e ciência coletiva do código. O *Bitbucket* fornecerá o controle de versões ao

longo do desenvolvimento e o correspondente controle de acesso e autenticação ao código. O *Travis CI*, por sua vez, dará suporte à integração e à comunicação entre as ferramentas supracitadas.

System Metaphor: as metáforas definidas e utilizadas pelos membros da equipe ao longo do desenvolvimento podem ser documentadas por meio de editores de texto, ferramentas gráficas, planilhas etc. Uma alternativa é o uso de *wikis* (como fornecida pelo *Bitbucket*) para documentar e compartilhar conhecimento sobre conceitos e relacionamentos do domínio da aplicação. Cada organização pode definir o melhor meio para o registro dessas metáforas.

40-hour week: como parte do planejamento, essa prática está ligada diretamente ao *Taiga* e ao *Slack*. Por meio dessas ferramentas, pode-se acompanhar planejamento e execução das tarefas e promover melhor comunicação.

As ferramentas integradas no arcabouço proposto possuem interação entre si, permitindo, por exemplo, troca de informações e acionamento automático de funcionalidades. A Figura 4 ilustra tais interações, abordando tanto as ferramentas CASE na Nuvem quanto as locais. Tais interações são descritas a seguir.

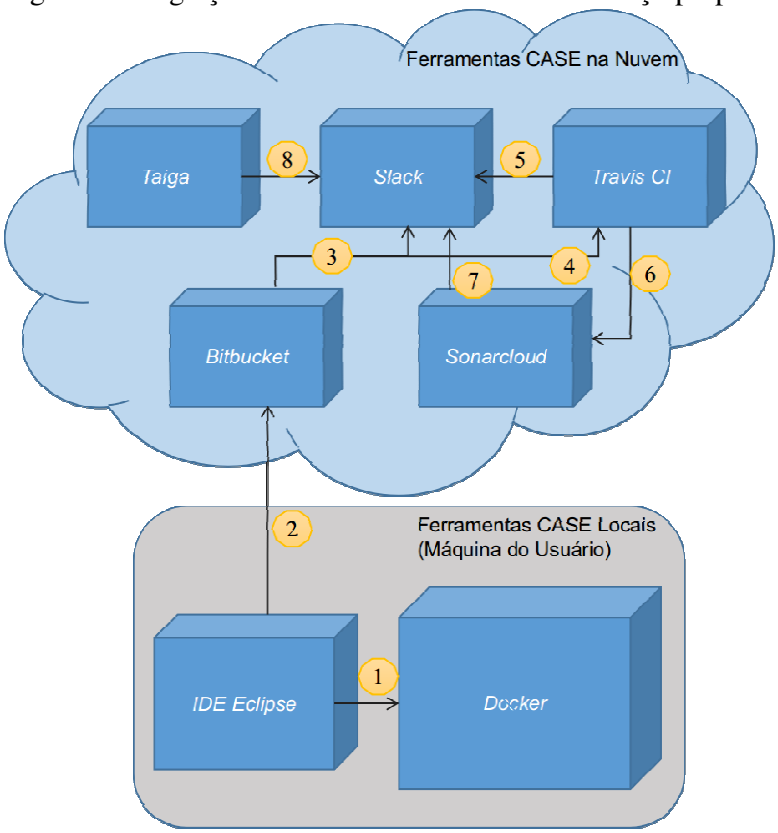
Interações

1. O *IDE Eclipse* estabelece comunicação com o *Docker,* a fim de acessar a infraestrutura de desenvolvimento (p.ex., banco de dados, bibliotecas e *frameworks* de desenvolvimento).
2. O *IDE Eclipse* acessa e/ou envia arquivos para o Bitbucket atualizando o repositório de arquivos.
3. O *Bitbucket* aciona o *Slack* para que esse notifique os envolvidos no projeto a respeito das alterações no repositório;
4. O *Bitbucket* aciona o *Travis CI*, indicando os arquivos para realização do novo *build*.
5. Ao realizar o *build*, o *Travis CI* aciona o *Slack* a fim de manter os envolvidos no projeto informados sobre os resultados dos *builds*.
6. O *Travis CI* aciona o *Sonarcloud* para que esse realize a análise de qualidade do código.
7. O *Sonarcloud*, após realizada a análise do código, aciona o *Slack* para que esse notifique os envolvidos no projeto.
8. Por meio do *Taiga*, pode-se acionar o *Slack,* a fim de que esse também envie notificações acerca de mudanças no planejamento e execução do projeto.

ISSN 2359-4799
Volume 6 / Número 2 / Ano 2020 – p. 207-227
DOI: 10.36524/ric.v6i2.555

REVISTA Ifes ciência

Figura 4: Integração entre as ferramentas do arcabouço proposto



Fonte: Elaborado pelos autores.

Por fim, para evidenciar o atendimento aos requisitos descritos na Seção 3, o Quadro 3 relaciona cada requisito da solução com os respectivos aspectos tratados na solução proposta.

Quadro 3: Relação entre Requisitos e Aspectos Tratados na Solução Proposta

Requisitos	Solução Proposta
RQ1 – Arcabouço baseado em propósitos e tipos de ferramenta e não em ferramentas específicas.	O arcabouço foi baseado em um alinhamento genérico entre etapas do Scrum e práticas do XP.
RQ2 – Possibilidade de usar ferramentas CASE locais	Possível. Há ferramentas locais dentre aquelas já incorporadas ao arcabouço.
RQ3 – Ferramentas com mecanismos de integração entre elas.	Todas as ferramentas utilizadas possuem algum nível de integração (cf. descrito na Figura 4).
RQ5 – Base metodológica padrão que guie a utilização das ferramentas.	Processo base definido.
RQ6 – Solução baseada em Metodologias Ágeis (Scrum e XP).	Alinhamento entre Scrum e XP incorporado no processo base.

Fonte: Elaborado pelos autores.

6 TRABALHOS CORRELATOS

Nesta seção são apresentados alguns trabalhos correlatos, a fim de estabelecer uma comparação descritiva em relação ao trabalho proposto neste artigo.

Darwish (2014) propõe o ele chama de *enhanced Scrum framework*, o qual integra práticas do XP ao longo do processo do Scrum. O *framework* proposto por Darwish inclui um conjunto de diretrizes que buscam guiar a realização das atividades do Scrum. Tais diretrizes levam em consideração 8 (oito) das 12 (doze) práticas do XP. Por meio das diretrizes, o autor espera tornar o *framework* mais diretamente aplicável. Embora Darwish tenha focado apenas em um subconjunto das práticas do XP, o *framework* proposto por ele se relaciona a este trabalho na medida em que busca alinhar Scrum e XP com o objetivo de potencializar o uso dessas metodologias. Apesar das contribuições apresentadas por Darwish, vale descatar que seu trabalho não tem em seu escopo o uso da Computação em Nuvem.

Qureshi (2017) propõe um modelo - XScrum -, que integra XP e Scrum em uma abordagem de desenvolvimento ágil. XScrum prevê 4 (quatro) fases em seu ciclo de desenvolvimento, a saber: (i) cenário (*plot*), (ii) padrão (*pattern*), (iii) instigar (*instigate*) e (iv) filtro (*filter*). Tais fases são mapeadas, de maneira ortogonal, no que o modelo define como atividades: (i) processo (*process*), (ii) comunicação e coordenação (*communication & coordenation*), (iii) garantia da qualidade (*quality assurance*), e (iv) satisfação do cliente (*client satisfaction*). Para cada relação de fase vs. atividade no XScrum são definidas práticas (muitas delas advindas do Scrum e do XP) a serem realizadas durante o processo de desenvolvimento de *software*. Em relação ao arcabouço proposto neste artigo, o modelo de Qureshi não estabelece um alinhamento direto entre Scrum e XP, mas o faz indiretamente pela proposição de um novo modelo, o XScrum. Neste trabalho, entretanto, decidiu-se pelo alinhamento direto entre XP e Scrum (sem proposição de um novo modelo) a fim de buscar promover a aplicação mais direta (já que Scrum e XP é de domínio de grande parte da comunidade ágil) evitando que colaboradores da indústria e da academia tivessem que aprender um terceiro modelo de desenvolvimento de software. Ademais, XScrum, diferente deste trabalho, também não estabelece aspectos em relação ao uso de ferramental CASE, seja local ou em Nuvem. Dessa forma, questões relacionadas à discussão Metodologias Ágeis e Computação em Nuvem não são tratadas no modelo proposto por Qureshi (2017).

Sever (2019) propõe um modelo cujo objetivo é facilitar a adaptação de metodologias ágeis para equipes distribuídas usando a Computação em Nuvem. O modelo é baseado em 4 (quatro) pilares do desenvolvimento ágil: (i) resposta dinâmica/rápida às mudanças, (ii) foco no produto (*software* funcionando), (iii) colaboração com o cliente e (iv) colaboração entre a equipe. No modelo, o processo de desenvolvimento é subdividido em três camadas que tratam: (i) do repositório do projeto (*process space*), (ii) dos modelos ágeis, do domínio e de inteligência (*model space*) e (iii) do processamento de decisão. Nesse sentido, o modelo busca incorporar outros aspectos, como tomada de decisão, para dar suporte ao processo de desenvolvimento de *software*. De maneira semelhante ao proposto neste artigo, Sever alinhou seu modelo a um conjunto de possíveis ferramentas em Nuvem a serem consideradas na instanciação/aplicação do modelo. No entanto, se comparado a este trabalho, o alinhamento das ferramentas é feito em um nível mais alto de especificidade/abstração, pois Sever não aborda uma metodologia ágil em particular. Ele relaciona as ferramentas CASE no nível em 4 pilares gerais de desenvolvimento ágil, considerados no seu modelo. Neste artigo, entretanto, as ferramentas CASE são diretamente mapeadas no nível de eventos e práticas do Scrum e XP, respectivamente. Tais questões são de fato aspectos de *design* de cada solução. Enquanto um modelo mais geral pode promover maior reuso, ele tende a exigir um maior esforço de aplicação em um cenário real. Por outro lado, um modelo com diretrizes mais específicas tende a um

menor reuso, mas pode ser mais diretamente aplicado. De todo modo, modelos mais gerais ou mais específicos podem coexistir e serem mutuamente úteis.

Younas *et al.* (2016) propõem um *framework* cujo objetivo é prover um ambiente para suporte ao desenvolvimento distribuído ágil de *software* usando Computação em Nuvem. O *framework* define 4 (quatro) passos a serem seguidos para seleção de ferramentas: (i) definição de metodologias ágeis e ferramentas de suporte, (ii) seleção de características baseadas na Nuvem, (iii) gerenciamento e repositório de código e (iv) comunicação e colaboração. De maneira análoga a Sever (2019), Younas *et al.* (2019) definiram seu *framework* considerando um nível maior de abstração, não focando explicitamente em uma metodologia ágil. Nesse sentido, o *framework* tende a ser mais genérico, apresentando uma visão mais ampla, mas, de certa forma, não entra em aspectos particulares de determinada metodologia, os quais poderiam ser mais diretos e específicos ao guiar o desenvolvedor ou gerente de equipe na adoção do modelo/*framework*. Ademais, o *framework*,proposto em Younas *et al.* (2019), apresenta uma estrutura em camadas com guias gerais para seleção das ferramentas. O arcabouço proposto neste trabalho, por sua vez, se baseia, primeiramente, em requisitos de ferramentas (baseados no alinhamento XP e Scrum) para que, a partir desses requisitos, se possa definir as melhores ferramentas.

7 CONSIDERAÇÕES FINAIS

Sistemas de *software* estão presentes no cotidiano da sociedade. Assim, a busca por *softwares* de qualidade que atendam a seus usuários e que sejam produzidos com eficiência é uma constante. Metodologias e ferramentas têm sido desenvolvidas e revistas no âmbito da Engenharia de *Software*.

Este trabalho tomou como cenário de pesquisa um laboratório de extensão universitária do Ifes, onde são desenvolvidos projetos de *software* para pequenas e médias empresas. A partir das questões relacionadas à falta de uma padronização metodológica e de um ferramental integrado e de fácil acesso, buscou-se propor um arcabouço que alinhasse o uso de metodologias ágeis - Scrum e XP - na definição de um processo base apoiado por ferramentas CASE em Nuvem integradas.

É importante ressaltar que, embora não se tenha realizado um experimento sistemático para avaliar a aplicabilidade e a eficácia da solução proposta, houve a oportunidade de apresentá-la a coordenadores e equipes de projetos do laboratório de extensão analisado. Pelo retorno positivo obtido, entende-se que a solução proposta é viável e tem potencial de aplicação em um cenário real.

Para além da solução proposta, este trabalho aborda aspectos historicamente importantes, associados ao uso de ferramentas CASE. Abordagens e discussões acerca da importância dessas ferramentas e, principalmente, do nível de integração requerido entre elas foram foco de várias pesquisas (BERTOLLO *et al.*, 2002; OLIVEIRA *et al.*, 2000; e SANTOS *et al.*, 2005). Agora, passa-se por um momento em que a disponibilização e uso de ferramentas CASE têm sido influenciados pela Computação em Nuvem. Assim, tem-se vivenciado uma evolução no uso dessas ferramentas passando de uma abordagem local (muitas vezes associada a uma *suíte* integrada *a priori* de ferramentas de um único fornecedor) para uma de uso em Nuvem, em geral, caracterizada por ferramentas heterogêneas disponibilizadas por diversos fornecedores e que necessitam ser integradas *a posteriori*. Tal transformação tem sido considerada com uma revolução no uso de ferramentas CASE e na maneira de construir *softwares* (ATKINSON, DRAHEIM, 2013).

Entende-se que essa mudança de abordagem foi possibilitada, dentre outros, pela disseminação e melhoria no acesso à Internet e pela evolução das tecnologias de

comunicação e integração de ferramentas de *software*. Acredita-se que essa nova tendência traz benefícios, na medida em que permite que organizações de *software* ampliem o suporte de ferramentas CASE ao processo de desenvolvimento por um custo mais competitivo e por uma independência maior de plataformas e fornecedores.

Como trabalhos futuros, espera-se incorporar novas ferramentas CASE em Nuvem ao arcabouço proposto como, por exemplo, ferramentas de modelagem conceitual (UML e Modelo Relacional) e ampliar o grau de automatização de tarefas. Ademais, espera-se aplicar o arcabouço em uma série de projetos reais, a fim de avaliá-lo, sistematicamente, relatando experiências e incorporando possíveis melhorias.

REFERÊNCIAS

ASHRAF, Sara; AFTAB, Shabib. Scrum with the Spices of Agile Family: A Systematic Mapping. **International Journal of Modern Education and Computer Science**, 2017, 11, pp. 58-72.

ATKINSON, Colin; DRAHEIM, Dirk. Cloud-Aided Software Engineering: Evolving Viable Software Systems Through a Web of Views. *In*: MAHMOOD, Zaigham; SAEED, Saqib (org.). **Soft. Engineering Frameworks for the Cloud Computing Paradigm**. Springer. 2013. pp. 255-281.

BECK, Kent. **Programação eXtrema (XP) eXplicada: Acolha as Mudanças**. Bookman. Porto Alegre. 2004.

BERTOLLO, G., *et al.* ODE - Um Ambiente de Desenvolvimento de Software Baseado em Ontologias. **Anais do XVI Simpósio Brasileiro de Engenharia de Software** - SBES'2002. Caderno de Ferramentas, pp.438-443, Gramado - RS, Brasil, Outubro 2002.

BENLIAN, Alexander; KETTINGER, William; SUNYAEV, Ali; WINKLER, Till. Special Section: The Transformative Value of Cloud Computing: A Decoupling, Platformization, and Recombination Theoretical Framework. **Journal of Management Information Systems**. 35. 2018, p. 719-739.

BUTT, Shariq A. **Study of agile methodology with the cloud**. Pacific Science Review: Humanities and Social Sciences 2. Elsevier. 2016, pp. 22 - 28.

BUYYA, Rajkumar *et al.* A manifesto for future generation cloud computing: Research directions for the next decade. **ACM Computing Surveys (CSUR).** 51.5. 105. 2018.

CASAGRANDE, L. M. **O que é Scrum?** Disponível em: http://apoemaconsultoria.com.br/o-que-e-scrum/. Acesso em: 20 maio 2018.

DARWISH, Nagy R. Enhancements in Scrum Framework Using Extreme Programming Practices. **Intern.** Journal of Intelligent Computing and Information Science. Vol.14, n. 2 April 2014.

FRANCO, E. F. **Um modelo de gerenciamento de projetos baseado nas metodologias ágeis de desenvolvimento de software e nos princípios da produção enxuta**. São Paulo: Biblioteca Digital de Teses e Dissertações da Universidade de São Paulo, maio, 2007.

HEVNER, A. R.; MARCH, S. T., PARK, J., and RAM, S., Design Science in Information Systems Research, ***MIS Quarterly2***, vol. 28, no. 1, 2004, pp. 75–105.

KRISHNA, Radha; JAYAKRISHNAN, R. Impact of Cloud Services on Software Development Life Cycle. *In*: MAHMOOD, Zaigham; SAEED, Saqib (org.). **Soft.** Engineering Frameworks for the Cloud Comp. Paradigm. Springer. 2013. pp.79 - 99.

KARUNAKARAN, Sowmya. Impact of Cloud Adoption on Agile Software Development. *In*: MAHMOOD, Zaigham; SAEED, Saqib (org.). **Soft. Engineering Frameworks for the Cloud Computing Paradigm**. Springer. 2013. p. 255 - 281.

OLIVEIRA, K. M. *et al.* Construção de Ambientes de Desenvolvimento de Software Orientados a Domínio na Estação TABA, **Memórias de Jornadas Iberoamericanas de Ingeniería de Requisitos y Ambientes de Software**, IDEAS'2000, Cancún, Mexico, Julho 2000.

MARCONI, M. DE A.; LAKATOS, E. M. **Fundamentos de Metodologia Científica**. 5. ed. v. 29. São Paulo: Editora Atlas, 2003.

MUPPALLA, Anil K.; PRAMOD, N.; SRINIVASA, K. G. Efficient Practices and Frameworks for Cloud- Based Application Development. *In*: MAHMOOD, Zaigham; SAEED, Saqib (org.). **Soft. Engineering Frameworks for the Cloud Computing Paradigm**. Springer. 2013. pp. 305 - 329.

QURESHI, M. Rizwan J. Evaluating the Quality of Proposed Agile XScrum Model. **I.J. Modern Education and Computer Science**, 11, 2017, pp. 41-48.

SAHMIM, Syrine; GHARSELLAOUI, Hamza. **Privacy and Security in Internet-based Computing:** Cloud Computing, Internet of Things, Cloud of Things: a review. International Conference on Knowledge Based and Intelligent Information and Engineering Systems, KES2017, 6-8 September 2017, France.

SANTOS, G. *et al.* **Knowledge Management in a Software Development Environment to Support Software Processes Deployment**. Professional Knowledge Management, Third Biennial Conference, WM 2005, Kaiserslautern, Germany, April, 2005, pp. 10-13.

SABBAGH, R. **Scrum Gestão Ágil para projetos de Sucesso**. 1. ed. Rio de Janeiro: Editora Casa do Código, 2014.

SCHWABER, K.; SUTHERLAND, J. **Guia do Scrum:** As regras do jogo. Disponível em:https://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-Portuguese-BR.pdf. Acesso em: 11 mar. 2018.

SEVER, Ali. Modeling Distributed Agile Software Development Utilizing Cloud Computing: A Holistic Framework. **Current Journal of Applied Science and Technology**. 35 (6): 2019, pp. 1-12.

SILVA, D. E. DOS S.; SOUZA, I. T. DE; CAMARGO, T. Metodologias Ágeis Para O Desenvolvimento De Software: Aplicação E O Uso Da Metodologia Scrum Em Contraste Ao Modelo Tradicional De Gerenciamento De Projetos. **Revista Computação Aplicada**, 2, 1, 2013, pp. 39–46.

SOARES, M. DOS S. Metodologias Ágeis Extreme Programming e Scrum para o Desenvolvimento de Software. **Revista Eletrônica de Sistemas de Informação**, v. 3, 2004, pp. 1–8.

SOMMERVILLE, I. **Engenharia de Software**. Addison Wesley Bra. 2007.

SUNYAEV, Ali. **Internet Computing:** Principles of Distributed Systems and Emerging Internet-Based Technologies. Springer. February, 2020.

TRITSINIOTIS, Emmanouil. **Get ready for the Cloud:** Tailoring Enterprise Architecture for Cloud Ecosystems. LAP LAMBERT Academic Publishing. MASTER OF SCIENCE THESIS. Enschede University, 2013.

LIZHE, W.; JIE T.; MARCEL, K.; ALVARO C. C.; DAVID K.; WOLFGANG K. **Scientific Cloud Computing:** Early Definition and Experience. 10th IEEE Intern. Conference

on High Performance Computing and Communications. Dalian, China. 2008.

YOUNAS, Muhammad; GHANI, Imran; JAWAWI, Dayang N. A.; KHAN, Muhammad M. A Framework for Agile Development in Cloud Computing Environment. **Journal of Internet Computing and Services (JICS)**. Oct.: 17(5): 2016, pp. 67-74.

YOUNAS, Muhammad; JAWAWI, Dayang N.A.; GHANI, Israr; KHAN, Muhammad I.; GHANI, Imran. A survey of agile development methods and tools in cloud environment. **Journal of Software Engineering & Intelligent Systems**, Volume 2, Issue 3. 2017, pp. 236-242.

YOUNAS, Muhammad; JAWAWI, Dayang N.A.; GHANI, Imran; FRIES, Terrence, KAZMI, Rafaqut. Agile development in the cloud computing environment: A systematic review. **Information and Software Technology**, v. 103, 2018, pp.142-158.