CONTINUOUS INTEGRATION AND VERSION CONTROL: A SYSTEMATIC REVIEW

Conference Paper · May 2017

DOI: 10.5748/9788599693131-14CONTECSI/RF-4435

5 authors, including:

Fabio Rocha
Institute of Technology and Research
45 PUBLICATIONS **77** CITATIONS
SEE PROFILE

Pablo Marques Menezes
Universidade Tiradentes
9 PUBLICATIONS **2** CITATIONS
SEE PROFILE

Rogerio Patricio Chagas do Nascimento
Universidade Federal de Sergipe
79 PUBLICATIONS **105** CITATIONS
SEE PROFILE

Methanias Colaço Júnior
FEDERAL UNIVERSITY OF SERGIPE (UFS)
90 PUBLICATIONS **141** CITATIONS
SEE PROFILE

DOI:10.5748/9788599693131-14CONTECSI/RF-4435

CONTINUOUS INTEGRATION AND VERSION CONTROL: A SYSTEMATIC REVIEW

Fabio Gomes Rocha (Universidade Tiradentes, Sergipe, Brasil) – gomesrocha@gmail.com
Pablo Marques Menezes (Universidade Tiradentes, Sergipe, Brasil) – pmenezes.aju@gmail.com
Rogério Patrício Chagas do Nascimento (Universidade Federal de Sergipe, Sergipe, Brasil) – rogerio@ufs.br
Methanias Colaço Rodrigues Junior (Universidade Federal de Sergipe, Sergipe, Brasil) - mjrse@hotmail.com
Colaborator:
Adicinéia Aparecida de Oliveira (Universidade Federal de Sergipe, Sergipe, Brasil) - adicineia@gmail.com

Fast delivery of functional parts has been one of the most complex challenges in the software development process. This article aims at using a systematic review of literature between 2005 and 2015, to identify the state of the art on the technical continuum integration and tools of version control, Subversion and Git, identified as the most used. The works found are analyzed and presented in this article

Keywords: Continuous Integration, Version Control, Continuous Delivery, Git, SubVersion

INTEGRAÇÃO CONTÍNUA E CONTROLE DE VERSÃO: UMA REVISÃO SISTEMÁTICA

A entrega rápida de partes funcionais tem sido um dos mais complexos desafios no processo de desenvolvimento de software. Este trabalho objetiva identificar o estado da arte sobre a técnica de integração contínua e as ferramentas de controle de versões, Subversion e Git, identificadas como as mais utilizadas, utilizando técnicas de revisão sistemática na literatura, entre 2005 e 2015. Os trabalhos encontrados são analisados e apresentados neste artigo, levando a conclusão de que não há uma metodologia ou ferramenta amplamente utilizada na integração contínua entre os trabalhos analisados

Palavras-chave: Integração Contínua, Controle de Versão, Revisão Sistemática, Git, Subversion.

1 Introdução

Desde a década de 1950 o desenvolvimento de software vem passando por mudanças e sendo alvo de estudos (LARMAN, BASILI, 2003). A forma em que as equipes desenvolvem e entregam produtos sofrera mudanças estruturais. Em 1986, Barry Boehm criou o modelo espiral empregando processos de desenvolvimento iterativo e incremental (LARMAN, BASILI, 2003). Isso aumentou a necessidade do gerenciamento de configuração de software, devido ao desenvolvimento por partes em cada fase do processo.

Entre as funcionalidades da gerência de configuração, que é o conjunto de tarefas para manter sistemas, o controle de versão é considerado um dos elementos chaves para o processo(HUMBLE, FARLEY, 2014), pois busca minimizar os problemas advindos do ciclo de vida do software por meio de controles sistêmicos das mudanças. Assim, surgiu, ainda na década de 1970, o Source Code Control System (SCCS) como ferramenta Unix proprietária(ROCHKIND, 1975). Posteriormente, buscando melhorar o processo de gestão de código fonte, em 1983, Tichky cria uma alternativa ao SCCS denominada Revision Control System ou RCS (TICHY, 1984). A limitação apresentada pelo RCS era o gerenciamento apenas de arquivos, não envolvendo projetos inteiros. A demanda pelo controle de versões de projetos inteiros fez surgir, em 1986, o Concurrent Version System (CVS), originalmente criado por Dick Grune (HUMBLE, FARLEY, 2014).

O CVS possuía diversas restrições, como o rastreamento apenas do histórico de arquivos individuais, o versionamento somente de arquivos, entre outras. Isso levou a empresa CollabNet criar uma alternativa, cujo foco era manter a compatibilidade com o CVS adicionando outros recursos: como a criação de um sistema de arquivos virtual para o controle de versão, um histórico de versões efetivos etc. Em agosto de 2001, tornando-se auto gerenciável, surgiu o Subversion (PILATO, COLLINS-SUSSMAN, FITZPATRICK, 2011), que foi projetado para ser um CVS de maior qualidade, mas mantendo a mesma estrutura de comandos. Com isso, o Subversion tornou-se popular rapidamente.

Como o Subversion herdava características do CVS, Linux Torvalds criou, em 2005, o Git, um sistema de controle de versão distribuído, visando controlar as versões do Kernel do Linux. A sua popularidade ocorreu nos anos seguintes, trazendo consigo novos conceitos e características não herdadas do CVS. O Git tem o foco em velocidade e designer simplificado, estando, juntamente ao Subversion, entre os sistemas mais utilizados para gerenciar versões no mundo (HUMBLE, FARLEY, 2014).

O controle de versão tem a tarefa de manter sistemas de software, que consistem em muitas versões e configurações, bem organizados. Ele permite que sejam acompanhadas as mudanças ao longo do tempo, rastreando cada alteração feita no software. Como resultado, é possível recriar uma cópia consistente em qualquer ponto no tempo (BALL, PORTER, SIY, 1997). Dessa forma, o controle de versão permite detectar conflitos de atualização de arquivos, evitando alterações que sobrepõem arquivos em funcionamento (TICHY, 1984). Caso tenham ocorrido mudanças incorretas, também é possível retornar as versões antigas do sistema e ou arquivos. Esse controle de versão não deve ser utilizado apenas para gerenciar código fonte. Tudo o que for necessário para a compilação, implantação, teste e entrega da aplicação (incluindo sua documentação, scripts de testes, scripts de configuração, etc.) devem ser gerenciados pelo sistema de controle de versão. O seu objetivo é armazenar cada versão dos arquivos desenvolvidos e garantir o acesso a eles (HUMBLE, FARLEY, 2014). Observa-se, assim, que os Sistemas de Controle de versão são o centro da entrega continuada, tornando-se necessários ao funcionamento dos sistemas de integração continuada.

Embora Humble e Farley (2014) afirmem que Kent Beck foi o primeiro a tratar sobre

"integração continuada", o artigo de Booch intitulado Coming of Ageinan Object-Oriented World, publicado em 1994, já trazia o conceito sobre o termo (BOOCH, 1994). Mas foi com a metodologia de desenvolvimento de software XP que a técnica veio a se popularizar, com a afirmação de que a integração ocorre imediatamente após o desenvolvimento (BECK, 2008).

A integração contínua é uma forma de garantir que todos possam facilmente acompanhar o estado do sistema e de suas mudanças (FOWLER, 2015). Por meio dela descreve-se um conjunto de práticas da engenharia de software que aceleram a entrega de produto, reduzindo o tempo de integração, criando oportunidade para reconhecer riscos mais cedo e permitindo correções incrementais sem desestabilizar o esforço da equipe de desenvolvimento (BOOCH, 1994).

O objetivo deste artigo foi, por meio de uma quasi-revisão sistemática, estudo do tipo secundário, abrangendo o período de 2005 a 2015, sumarizar os resultados, identificando o estado da arte da integração continuada, em conjunto com o controle de versão. Para isso, foram selecionadas as duas ferramentas mais utilizadas: Subversion com 47% de usuários, e Git com 38% (DUCK, 2015). O ano de 2005 foi escolhido como inicio devido o fato de ser o ano da criação do Git.

O trabalho está organizado da seguinte forma: na seção 2 é apresentado o protocolo de revisão sistemática; na seção 3 são expostos os conceitos de integração contínua; na seção 4 são abordados os conceitos de controle de versão, na seção 5 são tratados os resultados da revisão sistemática; na seção 6 apresenta-se um resumo dos trabalhos selecionados; e, na seção 7, faz-se uma análise sobre os resultados e por fim são apresentadas as considerações finais.

2 Protocolo de revisão

Esta seção apresenta o protocolo empregado na pesquisa para a aplicação da revisão sistemática, realizada com o apoio da ferramenta Docear (2015), a qual é integrada ao JabRef (2015). Essa ferramenta permitiu o gerenciamento de referências. O objetivo da pesquisa foi formalizado usando parte do modelo GQM (BASILI, WEISS, 1984; VAN, BERGHOUT, 1999):

- Analisar: as publicações do IEEE e ACM;
- Com o propósito: de caracterizar;
- Com respeito: à integração continua empregando controle de versão utilizando Subversion ou Git;
- Do ponto de vista: de pesquisadores internacionais;
- No contexto de: pesquisas teóricas e aplicadas.

Tal objetivo está norteado pela seguinte questão:

Q1. Quais abordagens mais utilizadas nos artigos de integração continua e controle de versão?

A revisão foi realizada em três etapas: a primeira de forma exploratória, com o objetivo de analisar o caminho para que fosse possível encontrar as palavras-chaves e criar a strings de busca; na segunda etapa, foi realizado a pesquisa sem limitar período de tempo, identificando o inicio das publicações sobre o assunto em 1984; e na terceira etapa, já empregando os critérios de seleção o processo de revisão sistemática teve como objetivo identificar quais as técnicas e ou abordagens empregadas para a integração contínua, em conjunto com o controle de versão.

2.1 Seleção dos estudos

Os critérios de seleção de fontes foram:

- consultas de artigos através da Web;
- emprego de mecanismo de busca através de palavras chave e
- identificação de artigos publicados entre 2005 e 2015.

Critérios de inclusão foram:

- os artigos devem estar escritos em inglês e português;
- os artigos devem estar disponível no IEEE ou ACM;
- textos completos de estudo em formato eletrônico;
- os artigos devem apresentar estudos sobre Integração Continuada, empregando controle de versão com uma das ferramentas: Subversion ou Git;
- o título dos artigos deve possuir referência à integração, integração continua ou controle de versão.

O critério de exclusão definido foi "não deve existir artigos que não levantem os problemas relativos à integração contínua".

2.2 Busca e extração de dados

As consultas foram realizadas na base do IEEExplorer (2015) e ACM Digital Library (2015) por serem consideradas as principais bibliotecas digitais da área de computação. As strings de busca utilizadas foram:

1. (((Continuous Integration) AND Version Control System) AND ((Git) OR (Subversion))
2. (((Integração Continuan) AND Sistema de Controle de Versão) AND ((Git) OR (Subversion))

As buscas foram realizadas por dois pesquisadores, de forma independente, sobre trabalhos publicados nas fontes. Após a conclusão desta etapa, os pesquisadores debateram para obterem consenso sobre uma seleção dos estudos a serem analisados.

3 Integração Contínua

A integração contínua não é apenas uma ferramenta, mas uma mudança de paradigma, permitindo que a equipe entregue software funcionando constantemente ou continuamente, aspecto que se constitui em um dos princípios do manifesto ágil. O processo de integração se constitui em um desafio constante no desenvolvimento de softwares, principalmente com o aumento da complexidade dos programas. Há, cada vez mais, uma maior necessidade de integrar e garantir que o software e seus componentes trabalhem em conjunto (DURVALL, MATYAS, GLOVER, 2007; MEYER, 2014; ROCHA, 2015).

A integração contínua garante feedback preciso e frequente sobre o progresso de desenvolvimento, sendo essencial para diversas metodologias ágeis, como por exemplo a XP. Ela é indispensável para as entregas frequentes, possibilitando a liberação de versões em ciclos curtos de implementações. Assim, o código é integrado após algumas horas ou, no máximo, em um dia durante o desenvolvimento (BECK, 2008).

Novos códigos devem ser integrados constantemente. Nenhum código deve permanecer fora da integração por mais de algumas horas. A melhor forma para isso ocorrer é a integração imediata ao sistema a cada mudança que a equipe realizar (BECK, 2008). Dessa forma, as ferramentas devem promover a integração com o sistema de controle de versão e efetuar as builds, a partir do repositório, executando testes automatizados (PRIKLADNICKI, WILLI, MILANI, 2014).

Para uso da integração contínua, todo o código deve ser mantido no repositório de controle de versão. Assim, quando a equipe atualizar o repositório, o sistema de integração, de forma automatizada, coletará as mudanças efetuadas, verificará o código e executará um conjunto de ações de validação relativas as mudanças, buscando uma união ao sistema com

o mínimo possível de problemas. Essa ferramenta, portanto, torna-se um juiz imparcial para julgar se a mudança funciona. Será considerado impedido que um novo recurso, funcional no computador do desenvolvedor, entre em conflito com o sistema principal (MEYER, 2014). Consolida-se, assim, a necessidade de uso de ferramenta automatizada, pois problemas de integração ocorrem. Embora um código estaja operacional em âmbito individual, podem ocorrer problemas no momento da união ao sistema. Portanto, uma ferramenta que valide se uma mudança realmente se integra e funciona adequadamente, reduz o estresse e facilita o trabalho da equipe.

As equipes que empregam sistemas de integração contínua obtêm feedback constante sobre a integração do sistema, podendo receber informações por diferentes meios, como o dashboard da aplicação ou e-mail. Ao receber a informação, os desenvolvedores podem corrigir os problemas com maior agilidade, utilizando o sistema de controle de versão para atualizar o sistema.

4 Controle de Versão (Subversion e Git)

Desenvolver sistemas é mais do que apenas codificar em linguagens de programação, envolve documentação, scripts de instalação, etc. Tudo o que for necessário para o funcionamento da aplicação deve ser gerenciado pelo Sistema de Controle de Versão (SCV). O SCV é o mecanismo que possibilita a colaboração entre pessoas envolvidas na entrega do produto, podendo ser definido como um local para armazenamento, controle de acesso e mudanças de artefactos (MASON, 2006). As duas funções básicas do SCV são guardar as versões do arquivo, garantindo que seja possível realizar uma restauração, ou acesso a ele, e permitir que equipes distintas possam colaborar (HUMBLE, FARLEY, 2014).

O Sistema de controle de versão permite e facilita o trabalho paralelo de pessoas em um mesmo projeto, com a aplicação de manutenção e melhorias nos sistemas, o compartilhamento de documentação e a atualização para todos os membros do time. Observa-se, então, que o Sistema de controle de versão é uma ferramenta central para o sucesso de um projeto, facilitando o trabalho de equipes. Além disso, ele possibilita a integração de diversas ferramentas para capturar informações e métricas da equipe para que seja possível efetuar o trabalho.

5 Resultados da revisão sistemática

A execução do protocolo de revisão sistemática, na segunda etapa sem filtro de período, resultou em 91 artigos na base do IEEE e 31 na base da ACM, totalizando 122 artigos, sendo o primeiro publicado em 1984, conforme pode ser visto a o gráfico 1.

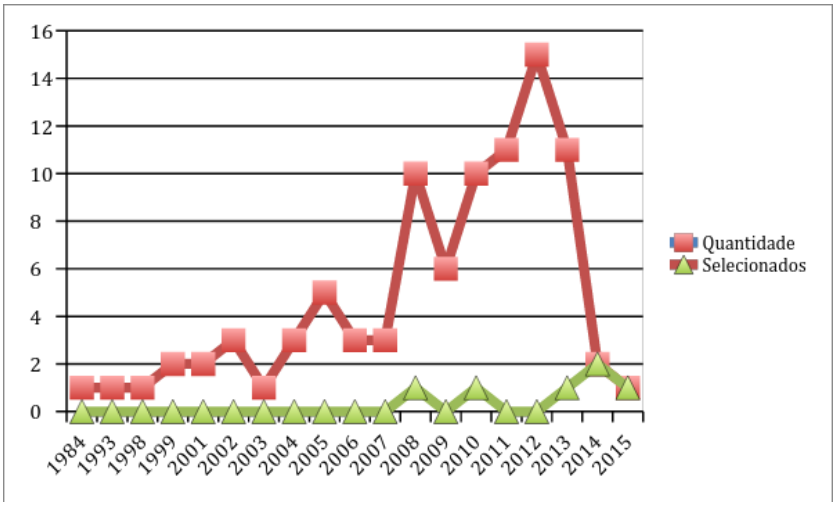

Gráfico 1. 1. – Gráfico de artigos por ano IEEE e ACM

O refinamento para a busca de artigos publicados apenas nos últimos 10 anos, devido ao fato de em 2005 ser o ano de criação do Git (HUMBLE, FARLEY, 2014), resultou em 77 trabalhos no IEEE, sendo possível observar uma expansão de publicação sobre o tema entre os anos de 2005 a 2015, com o ápice em 2012, representando 16,48% dos artigos publicados nesta base. Porém, não houve mudanças no numero de artigos na ACM, visto que os anos dos trabalhos vão de 2010 a 2015, totalizando 108 artigos nos últimos dez anos. Ao se somar o total entre 2005 e 2015, obteve-se o percentual de 88,524% de todos os artigos publicados, demonstrando a importância da pesquisa nesta área. Identificou-se, porém, uma queda a partir do ano de 2014, com apenas 2 artigos publicados no IEEE e 5 na ACM sobre o assunto.

O resumo de cada artigo foi lido, e os critérios de inclusão e exclusão foram aplicados. Após avaliação, foram selecionados sete artigos, sendo seis do IEEE e um da ACM para a leitura integral. Os resultados sobre a aplicação dos critérios de inclusão e exclusão dos artigos são expostos na tabela 1, a seguir.

Tabela 1 – Resultado quantitativo do protocolo de revisão

Métodos do protocolo	**ACM**	**IEEE**
Consulta *String* de busca	31	91
Período de 2005 a 2015	31	77
Critérios de inclusão	1	26
Critérios de exclusão	25	6
Seleção após a leitura dos resumos	1	6

Após a aplicação dos critérios de seleção e exclusão, foi realizada a leitura dos resumos, sendo selecionados um total de 7 artigos, (KIM, 2008; CALHAU, ALMEIDA FALBO, 2010; WALTERS, POO-CAAMAÑO,GERMAN, 2013; SHWETA, JOHN, SHENOY, 2014; WAITS, YANKEL, 2014; RAI, DHIR, GARG, 2015; GRUHN, HANNEBAUER,JOHN, 2013). O gráfico 2 apresenta a relação de artigos encontrados e selecionados, por ano de publicação a partir de 2005.

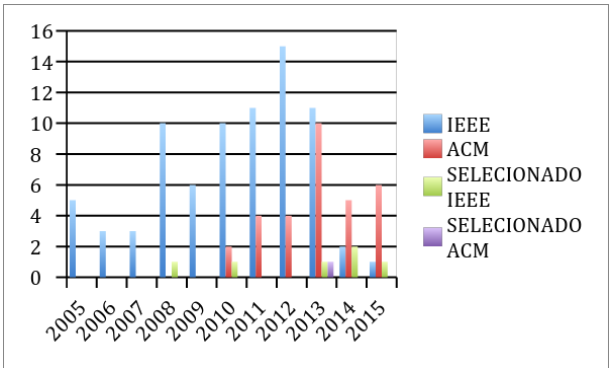

Gráfico 2. Relação de artigos encontrados e selecionados por ano.

6 Resumo dos trabalhos selecionados

O trabalho de Kim (2008) apresenta o procedimento de integração utilizado para um produto de software que possui centenas de componentes, além de integração de do sistema automatizado empregando a ferramenta Nightbird, utilizado no mesmo produto, porém com equipes distribuídas em três regiões sendo Korea, India e China. A abordagem utilizada segue alguns princípios de projetos que são: regras para o desenvolvedor, gerenciamento de componentes, gerenciamento da distribuição, adoção de construção de sistemas contínuos open source automatizados, teste automatizado de cada pacote, teste automatizado de todo o sistema e requisitos de performance. Assim com o uso do Nightbird para a distribuição de software composto de um grande número de pacotes e com as abordagens utilizadas no sistema de distribuição, garante-se a entrega de uma distribuição de produto com estabilidade.

Em outro trabalho, Calhau e Falbo (2010), apresentam uma abordagem a integração com uso de ontologias como modelos conceituais, mapeando os conceitos e serviços usados por diferentes aplicações empresariais. A proposta apresentada e denominada como Ontology-Based Approach for Semantic Integration (OBA-SI) é concentrada na análise de requisitos de integração e de modelagem. A integração é abordada em um alto nível de abstração e em três camadas: dados, serviços e processo. Como artefatos importantes, ela possui o domínio de referência ontológica que descreve o domínio de integração, os modelos conceituais dos aplicativos que estão sendo integrados e o modelo de processo do negócio que está sendo suportado pelas ferramentas. Com base em um estudo de caso, Calhau e Falbo (2010) utilizaram na preparação do ambiente as ferramentas Subversion (SVN) e Ontology-based software Development Environment (CM-ODE).

WALTERS, POO-CAAMAÑO e GERMAN, (2013) apresenta uma descrição do atual processos e distribuição de pacotes do projeto GNOME, tratando, ainda, sobre os dois problemas que esse modelo apresenta e trabalhos em curso que se destinam a resolver esses problemas. Os desafios descritos são a não possibilidade de haver várias versões de desenvolvimento de um sistema completo e a dificuldade de detectar regressões, pois essa atividade retarda o processo de garantia de qualidade. O documento ainda relata o uso da ferramenta OsTree que se integra ao controlador de repositório git-lite, utilizada no desenvolvimento, construção e implantação de sistemas baseados em Linux.

Já Gruhn, Hannebauer e John (2013) analisam a integração continua em projetos open source. Os autores verificaram que grande parte dos projetos possuem ambiente próprio, o que dificulta no processo de gerenciamento de segurança uma vez que a equipe, além de colaborar no projeto principal, deve manter o ambiente de integração de software. Ao final os autores apresentam uma prova de conceito indicando o Jenkins como uma solução

segura para integração contínua.

Já Shweta, John e Shenoy (2014) trata em seu trabalho sobre a otimização do processo de integração contínua. Os autores apontam, na seção de trabalhos correlatos, que esse tema é pouco discutido ou documentado. O trabalho propõe uma abordagem com o objetivo de alcançar maiores taxas de sucesso no processo de integração contínua, reduzindo substancialmente o tempo de verificação dos pedidos de desenvolvedor, e rápido feedback, aproveitando as capacidades de computação distribuída para atingir o objetivo. É descrito, ainda, um algoritmo aplicado na construção incremental de fontes. Esta abordagem propõe adicionar uma camada no processo de integração contínua, denominada Local Sanity Layer (LSL). Isso permite que as submissões de desenvolvedor sejam processadas pelo LSL antes de serem efetuadas. A LSL replica o servidor de integração contínua e realiza a integração das submissões, localmente. O trabalho apresenta resultados satisfatórios sobre o caso estudado, indicando uma melhora de 93 % no tempo de compilação para submissões enviadas pelos desenvolvedores onde ocorreram falhas na integração.

Com foco na documentação, Waits e Yankel (2014) apresentam o processo desenvolvido para gerir documentação nos formatos PDF e HTML, utilizando os recursos de integração contínua e controle de versão. Isso evitou o manuseio de arquivos binários. A integração do processo de documentação ao ambiente de desenvolvimento utilizou o texto plano e a linguagem de marcação. No entanto, essa abordagem foi utilizada para uma equipe pequena de desenvolvimento, o que acaba direcionando o desenvolvedor para ser o responsável pela documentação. O estudo demonstra que o processo de integração contínua pode ser aplicado ao processo de documentação com vistas ao aperfeiçoamento empregando a ferramenta Mercurial.

Por fim, Rai, Dhir e Garg (2015), apresentam e descrevem a ferramenta Jenkins para a integração contínua no desenvolvimento de software, na instalação e nas configurações básicas. Há ainda uma tabela comparativa entre o software tratado e outras ferramentas de integração contínua (Cruise Control - CC, Hudson, Apache's Continuum e Team City). Apesar do esforço em fazer um comparativo entre algumas ferramentas, não resta claro os parâmetros adotados, as referências sobre as informações apresentadas e o motivo da escolha dessas ferramentas. Pode-se analisar o fato de, em trabalhos futuros, verificar a estatística de uso de ferramentas de integração contínua no mercado e o motivo pelo qual elas serem as mais utilizadas.

7 Análise dos resultados

É possível observar que, embora não se apresente inovador, o tema conta com trabalhos publicados ainda na década de 1980, conforme pode ser observado na primeira busca sem os filtros de seleção por data. Porém, foi a partir de 2008 que o assunto recebeu maior atenção no meio acadêmico, com destaque em 2012.

Em resposta a pergunta sobre quais as abordagens mais utilizadas nos artigos de integração continua e controle de versão, temos que apesar de todos os artigos selecionados estarem dentro dos parâmetros de inclusão e contemplar os requisitos do protocolo de revisão sistemática, não estão circunscritos sobre o mesmo foco, dispersando-se entre:

- abordagem de integração empregando padrões;
- abordagem de integração com uso de ontologias;
- abordagem de integração utilizada para o projeto GNOME;
- abordagem de integração aplicado a gestão de documentos;
- otimização do processo de integração;

- abordagem comparativa entre as ferramentas.

Assim, apesar de todos os artigos tratarem sobre integração continua, não há alinhamento metodológico sobre as propostas, também não foi possível identificar ferramenta predominante para integração continua entre os artigos.

Cabe destacar ainda que um dos artigos emprega a integração contínua para gestão documental. Porém, a proposta foi aplicada a equipes pequenas, sendo necessário analisar o impacto em equipes maiores ou distribuídas. Outro aspecto relevante é que todos os trabalhos analisados, com exceção do artigo de Calhau e Falbo(2010) e Rai, Dhir e Garg (2015), são estudos com foco na aplicação prática empíricos, com aplicação em empresas ou projetos open source. Outro ponto importante, é que apesar do uso das ferramentas de controle de versão e integração contínua possibilitarem a extração de métricas de produtividade, nenhum dos trabalhos explorou este viés.

8 Considerações finais

A análise por meio de uma revisão sistemática apresentou o estado da arte da integração contínua, sumarizando resultados da pesquisa e indicando que os estudos estão sendo realizados sob metodologia empírica, havendo destaque para atuação em sistemas de código aberto (open source). Porém, não se identificou trabalhos sobre a sistematização do processo da integração contínua, bem como estudos de campo sobre os pontos positivos e negativos quando da utilização ferramentas de controle de versão e integração contínua. Os dados levantados indicam, apesar de ser um tema relativamente novo, que este vem progressivamente se expandindo entre os anos, demonstrando a importância de estudos sobre o assunto.

Apenas um trabalho apresenta técnicas de otimização do processo de integração contínua, o artigo de Shweta et al, e um expõe uma comparação sobre as ferramentas, o artigo de Rai et al, não apresentando, no entanto, um quadro comparativo, já o artigo de Gruhn, Hannebauer e John apresenta um estudo sobre a integração continua em projetos open source. Entre os demais, três artigos apresentam abordagens sobre o processo de integração e um aborda a integração contínua em conjunto com o controle de versão aplicado à gestão documental.

Dentro das circunstâncias nas quais essa pesquisa foi conduzida, observa-se, assim, não haver regularidade entre os trabalhos de integração contínua, não havendo uma metodologia ou ferramenta amplamente utilizada entre os trabalhos analisados, porém tanto o trabalho de Rai et al como no Gruhn, Hannebauer e John há indicação da ferramenta Jenkins para integração continua como solução segura para o processo, evitando-se a utilização de ambientes desenvolvidos internamente, evitando-se assim o estresse da equipe.

Apesar da importância de se medir a qualidade do produto e da produtividade da equipe, o que pode ser feito através da extração de conhecimento das ferramentas de controle de versão e de integração contínua, não foram encontrados trabalhos que apresentes tais resultados.

Para trabalhos futuros, é possível sugerir estudos de campo (survey) que caracterizem o uso das ferramentas de integração contínua e controle de versão. Sugere-se, ainda, um estudo mais amplo da aplicação da integração contínua no processo de entrega, bem como a avaliação sobre os benefícios e a identificação das limitações sobre as técnicas e processos de integração adotados atualmente. Outra linha é pesquisar como extrair métricas de produtividade e de qualidade através das ferramentas de controle de versão e integração contínua.

Este trabalho tem como principal contribuição, apresentar um estado da arte sobre integração continua, demonstrando pontos que demandam pesquisas para a melhor compreensão.

9 Referencias

ACM Digital Library. (2015). Disponível em < ACM Digital Library>.

Ball, T., Kim, J. M., Porter, A. A., & Siy, H. P. (1997, May). If your version control system could talk. In ICSE Workshop on Process Modelling and Empirical Studies of Software Engineering (Vol. 11).

BASILI, V.; WEISS, D. (1984, nov). A Methodology for Collecting Valid Software Engineering Data. In: IEEE TRANSACTIONS ON SOFTWARE ENGINEERING. vol.10(3): 728-738.

BECK, Kent. (2008). Programação Extrema (XP) explicada: Acolha as mudanças. Porto Alegre: Bookman.

BOEHM, B. (2006, May). A view of 20th and 21st century software engineering. In Proceedings of the 28th international conference on Software engineering (pp. 12-29). ACM.

BOOCH, Grady. (1994). Coming of Age in an Object-Oriented World. IEEE Software, Vol. 11, No. 6, p. 33-41.

CALHAU, Rodrigo Fernandes; FALBO, Ricardo de Almeida. (2010). An ontology-based approach for semantic integration. In: Enterprise Distributed Object Computing Conference (EDOC), 14th IEEE International. IEEE, 2010. p. 111-120.

Collins-Sussman, B., Fitzpatrick, B., & Pilato, M. (2004). Version control with subversion. O'Reilly Media Inc.

DOCEAR. (2015) Disponível em < http://www.docear.org/ >.

DUCK, Black. (2015). Compare Repositories. Disponível em: < https://www.openhub.net/repositories/compare>.

DURVALL, Paul M.; MATYAS, Steve; GLOVER, Andrew. (2007). Continuous Integration: Improving Software Quality and Reducing Risk. Boston(US): Addison-Wesley.

FOWLER, Martin. (2015). Continuous Integration. Disponível em: < http://www.martinfowler.com/articles/continuousIntegration.html>.

GRUHN, Volker; HANNEBAUER, Christoph; JOHN, Christian. (2013). Security of public continuous integration services. In: Proceedings of the 9th International Symposium on Open Collaboration. ACM. p. 15.

HUMBLE, Jez; FARLEY, David.(2014) Entrega Contínua: Como entregar software de forma rápida e confiável. Porto Alegre: Bookman.

IEEEXPLORER. (2015). Disponível em < ieee.org/ieeexplore>.

JABREF. (2015)Disponível em < jabref.sourceforge.net/ >.

KIM, Seojin et al. (2008). Automated continuous integration of component-based software: An industrial experience. In: Proceedings of the 2008 23rd IEEE/ACM International Conference on Automated Software Engineering. IEEE Computer Society. p. 423-426.

LARMAN, Craig; BASILI, Victor R. (2003). Iterative and incremental development: A brief history. Computer, n. 6, p. 47-56.

MASON, M. (2006). Pragmatic Version Control: Using Subversion (The Pragmatic Starter Kit Series). Pragmatic Bookshelf.

MEYER, Mathias. (2014). Continuous Integration and Its Tools. IEEE Software,

may/june, p. 14-16.

PRIKLADNICKI, Rafael; WILLI, Renato; MILANI, Fabiano. (2014). Métodos ágeis para desenvolvimento de software. Porto Alegre: Bookman.

RAI, Preeti; DHIR, Saru; GARG, Anchal. (2015). A Prologue of JENKINS with Comparative Scrutiny of Various Software Integration Tools. INDIACom.

ROCHA, Fabio. (2015). Integração Continua: uma introdução ao assunto. Disponível em < http://www.devmedia.com.br/integracao-continua-uma-introducao-ao-assunto/28002>

ROCHKIND, Marc J. (1975). The source code control system. Software Engineering, IEEE Transactions on, n. 4, p. 364-370.

SHWETA, M. A.; JOHN, Nimmy; SHENOY, Sneha. (2014). Improving Enterprise Build Process Using a Workflow Driven Approach in a Distributed Environment. In: Advances in Computing and Communications (ICACC), Fourth International Conference on. IEEE, 2014. p. 214-217.

Tichy, W. F. (1985). RCS—a system for version control. *Software: Practice and Experience*, *15*(7), 637-654.

VAN, S. R; BERGHOUT, E. (1999). The Goal/Question /Metric Method: A practical guide for quality improvement of software development. McGraw-Hill.

WAITS, Todd; YANKEL, Joseph. (2014). Continuous system and user documentation integration. In: Professional Communication Conference (IPCC), IEEE International. IEEE, 2014. p. 1-5.

WALTERS, Christine; POO-CAAMAÑO, Germán; GERMAN, Daniel M. (2013). The future of continuous integration in GNOME. In: Release Engineering (RELENG), 1st International Workshop on. IEEE, 2013. p. 33-36.