CONTINUOUS INTEGRATION AND VERSION CONTROL: A SYSTEMATIC REVIEW

Fabio Gomes Rocha - Tiradentes University, Pablo Marques Menezes - Tiradentes University, Rogrio Patrcio Chagas do Nascimento - Federal University of Sergipe, Methanias Colao Rodrigues Junior - Federal University of Sergipe, Adicinia Aparecida de Oliveira - Federal University of Sergipe

***Abstract*—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, Systematic Review, Git, Subversion.**

I. INTRODUCTION

Since 1950, software development has been undergoing changes and been studied [1]. The layout in which teams develop and deliver products had suffered structured changes. In 1986, Barry Boehm created the spiral model employing processes of interactive and incremental development [1]. It increased the need of software configuration management, due to the development in parts in each stage of the process. Among the features of the configuration management, which is the set of tasks to maintain the system, the version control is considered one of the key elements to the process [2], because it aims to minimize the problems related to softwares life cycle using systematic controls of changes. Thus arose still in the 70s the Source Code Control System (SCCS) as a Unix proprietary tool [3]. Subsequently, seeking to improve the source code manage process, in 1983, Tichky created an alternative to SCCS named Revision Control System or RCS [4]. The restriction presented by RCS was the management only for files, not involving full projects. The demand for control of full project versions has emerged, in 1986, the Concurrent Version System (CVS), originally created by Dick Grune [2]. The CVS had several restrictions as tracking only for individual file history, only file versioning, among others. It took the CollabNet company to create an alternative, focused in maintain the compatibility with CVS adding new features: as the creation of a new virtual file system for version control, a history of effective versions, etc. In August 2001, Subversion emerged becoming self-manageable [5] and projected to be a better quality CVS, but keeping the same command structure. Therefore, Subversion became popular very quickly. Since Subversion inherited CVS characteristics, Linux Torvalds created in 2005, the Git, a distributed version control system, aiming to control the Linuxs Kernel versions. Its popularity expanded in the following years, bringing with it new concepts and characteristics not inherited from CVS. Git focuses in speed and simplified design, being jointly to Subversion among the most used systems to manage versions in the world [2]. The version control has the task to maintain systems of software that consists in many versions and configurations well configured. It allows monitoring the changes through a long time, tracking each alteration made in the software. As a result, it is possible to recreate a consistent copy at any time [6]. That way, the version control allows detecting file update conflicts avoiding alterations that overlaps files in operation [4]. In case of incorrect changes have occurred, it is also capable of returning the old versions of the system or files. This version control should not be used only for manage source code. Everything necessary to compile, implement, test and application delivery (including its documentation, test scripts, configuration scripts, etc.) must be managed by the control version system. Its objective is to store each version of the developed files and to ensure the access to them [2]. It is observed that the version control systems are the center of the continuous delivery, becoming necessary to the operation of continuous integration systems. Although Humble and Farley [2] affirmed that Kent Beck was the first to deal with continuous integration, the article that Booch, called Coming of Ageinan Object-Oriented World, published in 1994 already brought the concept about the term (BOOCH, 1994). Only with XP software development that the technique came to be popular with the affirmation that integration occurs immediately after the development [7]. The continuous integration is a form of guarantee the easy accompaniment of the systems state and its changes [8]. In addition, he describes a set of software engineering practices that accelerates the product delivery, reducing the integration time, creating opportunities to acknowledge risks early and allowing incremental corrections without disturbing the team effort development [9]. The objective of this paper was, using systematic review, secondary type study, covering the period of 2005 to 2015, summarize the results, identifying the state-of-the-art of continuous integration together with version control. In order to get this, we selected the two most used tools: Subversion with 47% of users, and Git with 38% of user [10]. The year of 2005 was chosen as the starting point because it is Gits creation year. The paper was organized as follows. In the section 2, we present the protocol of systematic revision. The section 3 exposes the concepts of continuous integration. In the section 4, we approach the concepts of version control. In the section 5, there is the treated results of the systematic review. In the section 6, we present the summary of the selected papers. In the section 7, we build an analysis about the results and, finally, we present the final considerations.

II. Review Protocol

This section presents the protocol employed on the research to the systematic review application, accomplished with support of Docear [11] tool, which integrates JabRef [12]. This tool allowed the management of references. The objective of the research was formalized using part of the GQM model ([13]; [14]): - Analyze IEEE and ACM publications; - Purpose: characterize; - Respecting: to the continuous integration employing version control using Subversion or Git; - From a point of view: of international researchers; - In the context of: practical and theoretical papers. Therefore, the objective aims to the following question: Q1. Which are the most used approaches on continuous integration and version control articles? The review was conducted in three ways: the first one in an exploratory way, with the objective of analyze the path to find the keywords and create strings for research; the second way we conducted a research without limiting the period, identifying the beginning of publications about the subject in 1994; and lastly, employing the selection criterion, the systematic review had the objective of to identify the techniques and approaches used to continuous integration together with version control.

A. Selecting articles

The selection criterion of the sources was: - Consulting articles over the web; - Use of keywords in search mechanisms and; - Identifying articles published between 2005 and 2015. The criteria for inclusion was: - Articles should be written in English or Portuguese; - Should be available on IEEE or ACM; - Full texts in electronic format; - The articles must present studies about Continuous Integration, using Version Control with one of the tools: Subversion or Git; - The title of the articles must have the reference to integration, continuous integration or version control. The excluding criterion defined was: - There must be no articles that do not discuss the problems related to continuous integration.

B. Research and data extraction

We conducted the queries based on IEEExplorer [15] and ACM Digital Library [16] because they are considered as the main digital libraries for the area of computing. The strings for research used were: 1. (((Continuous Integration) AND Version Control System) AND ((Git) OR (Subversion))) 2. (((Integrao Contnua) AND Sistema de Controle de Verso) AND ((Git) OR (Subversion))) Two researchers realized the quests in an independent way, looking for papers published in the sources. After the conclusion of this round, the researchers debated to obtain a common sense about the selection of the studies to analyze.

III. Continuous Integration

Continuous integration is not only a tool, but is also the change of a paradigm, allowing the team to deliver the software running constantly and continuously, this aspect is a principle of the agile manifest. The process of integration consists in a constant challenge for software development, mainly with the increase of the programs complexity. There is a bigger need to integrate and ensure that the software and its components work together ([17]; [18]). The continuous integration guarantees precise and frequent feedback about the development progress, this is essential to many agile methodologies, as an example of XP. It is necessary to frequent delivery, making possible the release of versions in short cycles of implementation. Therefore, the code is integrated after a few hours or, at most, in one day during the development [7]. New codes must be integrated constantly. Every code should be integrated in few hours. The best way to make that happen is the immediate integration of the system at every change made by the team [7]. That way, the tools should promote the integration with the version control system and make the builds, from the repository, executing automatic tests [19]. For the use of continuous integration, every code should be kept in the repository of version control. That way, when the team update the repository, the integration system, automatically, will collect the changes made, verify the code and execute a set of validation actions related to the changes, searching for a system union with the minimum of problems. This tool, consequently, judges if the change works. It will be prevented a functional resource, at the developer computer, start a conflict with the main system [18]. We reinforce that, the usage of an automatic tool, because integration problems can happen. Although the code is running in an individual space, problems can happen in the moment of the system union. Therefore, a tool that validates if the change integrates and works successfully reduce the stress and facilitates the work team. The teams that use continuous integration systems obtain constant feedback about the systems integration, being able to collect information from different ways, as the application dashboard or e-mail. After receiving the data, the developers can correct the problems faster using the version control system to update the ambient.

IV. Version Control (Subversion and Git)

Development of systems is much mora than just code in programming languages, it involves documentation, installation scripts and etc. The Version Control System (VCS) should manage everything that is necessary to the correct working of the application. The VCS is the mechanism that enables the collaboration among the people involved with the product, it can be defined as a space to storage, access control and artifacts changes [20]. The two VCS basic functions are to keep the file versions, ensuring that is possible to realize restoration, or to give access to it, and to enable different teams to work together [2]. The version control system allows and facilitates the work of people in the same project, with the maintaining and improving of the systems, sharing documentation and updating for all members of the team. We observe that the version control system is the main tool to the success of the project, facilitating the teamwork. Besides that, it turns easy the integration of different tools for information gathering and team measure.

V. Results of the systematic review

The execution of the systematic review protocol, in the second stage without filters of time, resulted in 91 papers

on the IEEE base, and 31 on the ACM base, counting 122 papers, the first were published in 1984, as it can be seen Fig 1, orange symbolizes quantity and gray selected. . The

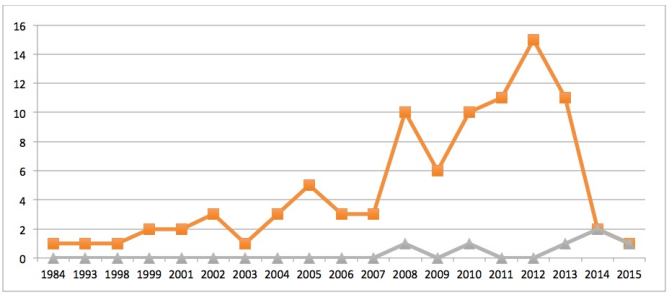

Fig. 1. IEEE and ACM graph of articles by year

refinement searching papers published in the last 10 years, due to the fact 2005 is the year of Gits creation [2], resulted in 77 articles in IEEE. Furthermore, we can observe the increase of publications about the topic between the years of 2005 and 2015, and the apex in 2012, representing 16.48% of the articles published in this base. However, there was no changes in the numbers of ACM articles, since the showing years goes from 2010 to 2015, numbering 108 papers in the last ten years. Counting the total between 2005 and 2015, we obtained the percentage of 88.524% of all published articles, proving the importance of research in this area. We also identified, also, a fall after 2014 year, with only two published articles in IEEE and five in ACM about the topic. We read the abstract of each paper and applied the inclusion and exclusion criteria. After evaluation, we selected seven articles, where six of them were from IEE and one from ACM for entire reading. The results about the application of the inclusion and exclusion criteria are exposed on the table 1:

TABLE I. TABLE 1: QUANTITATIVE RESULTS OF THE REVIEW PROTOCOL

Protocol methods	**ACM**	**IEEE**
Search string Consulting	31	91
Period of 2005 to 2015	31	77
Inclusion criterion	1	26
Exclusion criterion	25	6
Selection after abstract reading	1	6

After application of the excluding selection criterion, we read the abstracts, selecting a total of seven articles, ([21]; [22]; [23]; [24]; [25]; [26]; [27]). The Fig 2 shows the relation between the found articles and the selected ones for year of publication since 2005, being blue IEEE, ACM orange, gray selected IEEE articles and yellow selected ACM articles. .

VI. OVERVIEW OF THE SELECTED PAPERS

Kims paper [21] presents the integration procedure used for a product with hundreds of components as also the integration of an automatic system that uses Nightbirds tool, used in the same product even so the team was distributed in Korea,

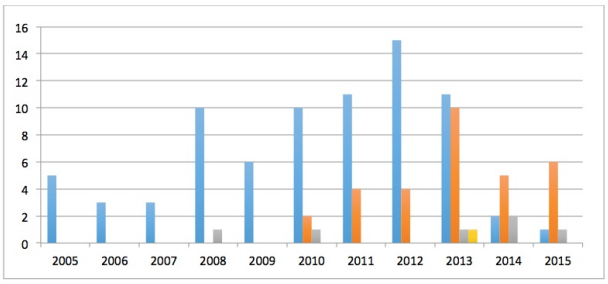

Fig. 2. List of articles found and selected per year.

India and China. The approach used requires some project principles: developer rules, component management, distribution management, construction of automatic open source continuous systems, automatic tests at each package, automatic tests in the whole system and performance requirements. That way, with the use of Nightbird for software distribution, composed by a large number of packages and approaches used for systems distribution, it ensures the delivery of a solid product distribution. In another paper, Calhau and Falbo [22], show an integration approach with the use of ontologies as concept models, mapping the concepts and services used by different business applications. The proposal submitted and named as Ontology-Based Approach for Semantic Integration (OBA-SI) is concentrated in a requirement analysis for integration and modelling. The integration is approached in a high level of abstraction and in three layers: data, services and process. As important artifacts, it has the domain of applications that are integrating and the business process model that the tools supports. Based in a study case, Calhau and Falbo [22] used, on the preparation of the ambient, the Subversion tools (SVN) and Ontology-based Software Development Enviroment (CM-ODE). WALTERS, POO-CAAMAO and GERMAN [23], present a description of the current process and distribution of the GNOME packages, treating the two process that this model present and works in operation that aims to solve those problems. The challenges described are about possibility of having many versions of development in a complete system and the difficulty of detecting regressions, because this activity slows the process of quality ensuring. The document still reports the usage of OsTree tool that integrates the git-lite repository controller, used on the development, building and implantation of Linux based systems. Gruhn, Hannebauer and John [27] analyze the continuous integration in open source projects. The authors verified that the large portion of the projects has their own ambient, that makes it difficult the security management process once the team have to keep the software ambient integrated and also collaborate with the main project. Finally, the authors present a concept indicating Jenkins as the solution for continuous integration. Shweta, John and Shenoy [24] discuss in their paper about the continuous integration process optimization. The authors point that this subject is not very discussed or studied. The paper proposes an approach with the objective of achieve better rates of success in the continuous integration process, reducing substantially the verification time

of the developer requests, getting faster feedbacks, using the capacities of distributed computing to reach the objective. It describes and algorithm applied on incremental construction of sources. This approach proposes to add a new layer on the continuous integration process, named Local Sanity Layer (LSL). It allows submissions from the developer that will be processed before being effectuated. The LSL reproduce the continuous integration server and make the integration of the submissions locally. The paper shows satisfactory results about the studied case, indicating an improvement of 93% of the time to compile sent submissions that had integration fails. Waits and Yankel [25] show the process developed to manage PDF and HTML documentation, using continuous integration and version control resources. It avoided the usage of binary files. The integration process of the development ambient documentation used a text of development that puts the developer to be responsible for the documentation. The study shows that the continuous integration process can be applied to the documentation process reaching the improving with Mercurial tool. Finally, Rai, Dhir and Garg [26], presented and described Jenkins tool for continuous integration on software development, installation and basic configuration. There is still a comparative table between the treated software and other tools for continuous integration (Cruis Controll CC, Hudson, Apaches Continuum and Team City). Despite significant efforts to compare a few tool, it is not clear the adopted parameters, the references about information showed and the why they chose these tools. In future works, we will analyze statistics about the markets usage of continuous integration tools and the reason why they are most used.

VII. Results analysis

We point that, even though it is not innovative, the subject has been studied since 1980, as we can see on the first search made without time filters. However, since 2008 the theme gained some importance in academic community, highlighting 2012 year. In response to the question about which approaches were more used on continuous integration and version control articles, although all the selected articles were within the inclusion parameters and they included systematic review protocol requirements, they have not the same focus, spreading in: - Different integration approaches standards; - Different integration ontologies usage; - Different integration used for GNOMES project; - Different integration applied to documents management; - Different optimization of the integration process; - Different comparative approaches of the tool. Although all the articles discussed continuous integration, they did not have any methodological alignment about proposals; also, we could not identify a predominant tool for continuous integration among the papers. The highlights also include that one of the papers use continuous integration for documental manage. However, the proposal was applied to small groups, so it is necessary to analyze the impact in larger or distributed groups. Another important aspect is that every one of the analyzed papers, excepting Calhau and Falbo [22] and Rai, Dhir and Garg [26], are studies focused on practical applications, as business companies or open source projects. We also emphasize that, although the usage of continuous integration and version control tools helped measure the production, none of it explored this theme.

VIII. Conclusion

The systematic review presented the continuous integration state-of-the-art, summarizing the research results and indicating that studies are being made under empiric methodology, highlighting their open source systems operation. However, we could not identify papers about systematization of continuous integration process, as well field surveys about positive and negative points when using continuous integration and version control tools. The data gathered indicate that, although it is a recent subject, which has been growing over the years, the study is a very important matter. Shweta et al was the only paper that presented techniques of process optimization, it also exposes a tools comparative, while Rai et al paper do not. Gruhn, Hannebauer and Jhon, presents a study about continuous integration in open source projects. Among these papers, three articles showed approaches about integration process and another one about continuous integration together with version control applied to documental manage. Under the circumstances of this research, we point that there is not a standard among the continuous integration papers. Not even a methodology, or a tool, strongly used in the analyzed papers. However, in Rai et al and Gruhn, Hannebauer and Johns papers have an indication of Jenkins tool for continuous integration as a secure solution for the process, avoiding using developed ambient internally and team stress. Although measuring the product quality and productivity of the team is very important, which can be done using continuous integration and version control tools; we did not find any paper that presented this study. For future papers, is possible to suggest a field survey that characterize the usage of continuous integration and version control tools. We suggest a larger study about the application of continuous integration on delivery process, as well the evaluation of the benefits, the identification of the techniques limits and integration process adopted nowadays. Another research line is the extraction of productivity and quality measures using continuous integration and version control tools. This paper has the objective of present the state-of-the-art about continuous integration, pointing that more studies are necessary for better comprehension.

Appendix A
Proof of the First Zonklar Equation

Some text for the appendix.

Acknowledgment

The authors would like to thank...

References

[1] C. Larman and V. R. Basili, "Iterative and incremental developments. a brief history," *Computer*, vol. 36, no. 6, pp. 47–56, 2003.

[2] J. Humble and D. Farley, *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation (Adobe Reader)*. Pearson Education, 2010.

[3] M. J. Rochkind, "The source code control system," *IEEE Transactions on Software Engineering*, no. 4, pp. 364–370, 1975.

[4] W. F. Tichy, "Rcsa system for version control," *Software: Practice and Experience*, vol. 15, no. 7, pp. 637–654, 1985.

[5] B. Collins-Sussman, B. Fitzpatrick, and M. Pilato, *Version control with subversion*. " O'Reilly Media, Inc.", 2011.

[6] T. Ball, J.-M. Kim, A. A. Porter, and H. P. Siy, "If your version control system could talk," in *ICSE Workshop on Process Modelling and Empirical Studies of Software Engineering*, vol. 11, 1997.

[7] K. Beck, *Extreme programming explained: embrace change*. addison-wesley professional, 2000.

[8] M. FOWLER, "Continuous integration," http://www.martinfowler.com/articles/continuousIntegration.html, Tech. Rep., 2017. [Online]. Available: http://www.martinfowler.com/articles/continuousIntegration.html

[9] G. Booch, "Coming of age in an object-oriented world," *IEEE Software*, vol. 11, no. 6, pp. 33–41, 1994.

[10] B. DUCK, "Compare repositories," https://www.openhub.net/repositories/compare, Tech. Rep., 2017. [Online]. Available: https://www.openhub.net/repositories/compare

[11] DOCEAR, "Docear," http://www.docear.org/, Tech. Rep., 2017. [Online]. Available: http://www.docear.org/

[12] JABREF, "Jabref," jabref.sourceforge.net, Tech. Rep., 2017. [Online]. Available: http://jabref.sourceforge.net

[13] V. R. Basili and D. M. Weiss, "A methodology for collecting valid software engineering data." NAVAL RESEARCH LAB WASHINGTON DC, Tech. Rep., 1983.

[14] R. Van Solingen and E. Berghout, *The Goal/Question/Metric Method: a practical guide for quality improvement of software development*. McGraw-Hill, 1999.

[15] IEEEXPLORER, "Ieeexplorer," ieee.org/ieeexplore, Tech. Rep., 2017. [Online]. Available: http://ieee.org/ieeexplore

[16] A. D. Library, "Dl acm," ACM, Tech. Rep., 2017.

[17] P. M. Duvall, S. Matyas, and A. Glover, *Continuous integration: improving software quality and reducing risk*. Pearson Education, 2007.

[18] M. Meyer, "Continuous integration and its tools," *IEEE software*, vol. 31, no. 3, pp. 14–16, 2014.

[19] R. Prikladnicki, R. Willi, and F. Milani, *Métodos ágeis para desenvolvimento de software*. Bookman Editora, 2014.

[20] M. Mason, *Pragmatic Version Control Using Subversion*. The Pragmatic Programmers LLC, 2006.

[21] S. Kim, S. Park, J. Yun, and Y. Lee, "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, 2008, pp. 423–426.

[22] R. F. Calhau and R. de Almeida Falbo, "An ontology-based approach for semantic integration," in *Enterprise Distributed Object Computing Conference (EDOC), 2010 14th IEEE International*. IEEE, 2010, pp. 111–120.

[23] C. Walters, G. Poo-Caamaño, and D. M. German, "The future of continuous integration in gnome," in *Proceedings of the 1st International Workshop on Release Engineering*. IEEE Press, 2013, pp. 33–36.

[24] M. Shweta, N. John, and S. Shenoy, "Improving enterprise build process using a workflow driven approach in a distributed environment," in *Advances in Computing and Communications (ICACC), 2014 Fourth International Conference on*. IEEE, 2014, pp. 214–217.

[25] T. Waits and J. Yankel, "Continuous system and user documentation integration," in *Professional Communication Conference (IPCC), 2014 IEEE International*. IEEE, 2014, pp. 1–5.

[26] P. Rai, M. Hooda, S. Dhir, M. Bhatia, and A. Garg, "A prologue of jenkins with comparative scrutiny of various software integration tools," in *Computing for Sustainable Global Development (INDIACom), 2015 2nd International Conference on*. IEEE, 2015, pp. 201–205.

[27] V. Gruhn, C. Hannebauer, and C. John, "Security of public continuous integration services," in *Proceedings of the 9th International Symposium on Open Collaboration*. ACM, 2013, p. 15.