Configuration Management to Tests Automatics in a Software Factory

Marcelo dos Santos Ferreira
C.E.S.A.R - Educational
Recife Center for Advanced Studies and Systems
Recife, Brazil
e-mail: marsantosfer@gmail.com

Ana Paula Cavalcanti Furtado
Informatics Center – C.I.n
Federal University of Pernambuco
Recife, Brazil
E-mail: anapaula.cavalcanti@gmail.com

***Abstract*—Artifact traceability during software development allows reusability and increases quality and productivity. Software requirements should be traceable because they represent the needs of a certain product. In several processes during software development cycle, requirements are related to other artifacts such as test suites and automation scripts. Configuration management assures the usage of an input file version corresponding to a software requirements version during automated tests. This research proposes an integrated process for software development, using configuration management from requirements up to test suites for automated tests. This article describes the effect of absence of configuration management on the control over software artifacts versions.**

Keywords-configuration management; reuse; tests automations; traceability; sotware requirements;

I. INTRODUCTION

The increasing demand for high quality products, made in shorter time, and at lower cost, requires from both academic world and industry innovative strategies and actions. Because software production is an industry of intangible goods, it has strong challenges in its production line as compared to conventional industry, which has fixed process input, tools, techniques and defined output [1]. Software production is tailor-made work, to fit client needs, with demands on functional and non- functional requirements. These requirements form input artifacts to several processes in a software factory.

During software development, the requirements might change due to variable market needs or legal regulation. Therefore artifacts traceability is necessary during the software building process [1].Configuration management assures artifacts traceability and keeps it in storage to control artifacts versions.

This article describes the impact of lack of configuration management on tools for test automation, and therefore legitimating the research on integrated process applied to software production.

This article has 5 sections, wherein the first section is an introduction. The motivation and problem definition are described in the second section. The third section addresses the methodology and its phases, and the proposed integrated process is discussed in the fourth section. Finally, in the fifth section, we have conclusion and future works.

II. LITERATURE REVIEW

Rework, low-quality products, demotivated teams, high software production and delays are symptoms caused by the absence of configuration management during software development, which directly affect users and development team [2]. This absence of configuration management creates the possibility for a software product does not meet the requirements.

The lack of version control of test suits and/or automatic test script to for the software requirements automation tests tools [1], reduces the reliability of such tools. In this way, software products can present nonconformities affecting directly the product acceptance by the client.

Configuration management (CM) should be active during whole software development, from infrastructure definition to information generation and maintenance. CM will support requirements changes during the development process resulting in flexibility during development. In addition to information and content of software requirements [3], the development process in software factories should perform requirements validations in all process sub-phases in order to guarantee continuous information and its understanding. Isolation of requirements after initiation phase can lead to wrong validation of content and its changes, consequently affecting the product that will be create.

The reusability of artifacts originated from software requirements, as well as source codes, tests suits and/or automatized scripts [4], is impaired by lack of relationship dependence between artifacts. This generates uncertainty concerning completeness of available artifacts, versions, and compatibility to project characteristics [5]. In this way, at each new automated test cycle, a new tests suite and/or automated scripts should be create to validate the most actual software requirements

Configurations management has only as unit control the configuration items [6] that are identified, controlled and kept, according the configurations management plan. The control realized in order to manage configurations does not include dependences and relationships between artifacts, in

 ISBN: 978-1-61208-498-5

other words, are isolated unit controls, without dependence relationship with other software artifacts. This configurations management approach has only a data and code reposition function.

The software tests that have as input software requirements to created test plans and tests projects are impaired by the lack of requirements traceability if requested changes are accepted included in requirements during development phase and the information is discontinued [2]. In this case, we might have scenarios where we will obtain software products having old versions of requirements. This affects products certification, as it does not meet their specifications. The tests automation tests when used in software process development [7], has as main function the coverage of regression tests and test suit. At one side, this will open the opportunity for the tests analysts to follow and test new features or to perform exploratory tests. On the other side, the use of automation test tools becomes a risk when tests suites and/or automatized scripts do not correspond to the actual version of software requirements. This will lead to loss of computation and human resources, which should be used to repair the missing update due to lack on artifacts traceability. The losses of computational and human resources reflect on delays in the product delivery and quality.

Based on these scenarios, we formulated the following research question: How to introduce a product-guided configuration management for automation tests in the development process of software factories?

Software requirements traceability was subject of study for Gotel [5] and Antoniol [3]. They realized interviews and surveys to identify support to requirements traceability. In addition, they identified problems related to providers or users of the software requirements. However, the research focus was limited to the requirement creation phase. Our research has the objective show how use configuration management [3] of software requirements and test suites for tests automatization tools will be able to solve several problems found in software factories.

III. RESEARCH METODOLOGY

The research approach was divided in four phases, as shown in Figure 2. The first phase corresponds to a literature review, that comprising bibliographical review and systematic review. In the second phase, we will postulate a proposal. Afterwards, we will evaluate the proposal with a case study and survey research. Finally, phase four should lead to suggestions for improvement.

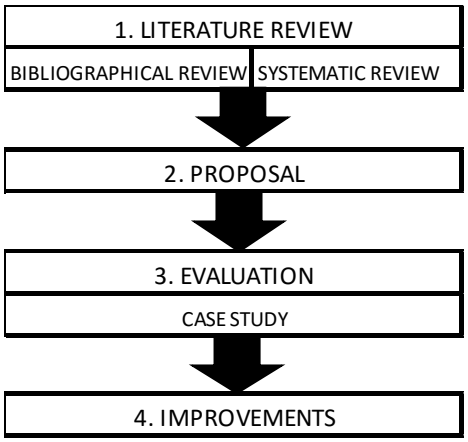

Figure 1. Schematic flow research activities.

A. Bibliographical Review

The review has as main objective the identification and exploration of scientific publications related to this study area. It will give an analysis of the previous studies in the field.

This review will be based on articles in the period of 2010 and 2015 available at IEEE digital libraries, ACM, Scopus, and Science Direct.

B. Systematic Review

The systematic review provides interpretation of relevant research in the field [8], followed by structure analysis [9] leading to gaps identification which might guide future new investigations in the field [10].

C. Case study

The case study has as main objective the analysis of a phenomenon within its context. The experimentations is an attempt to replication the phenomenon, taking into account factors, which affect software engineering results [11].

Software engineering involves development, operation, and maintenance of software and related artifacts [12]. The case study applies the research proposal and experimentation to determine the applicability of proposed solution, and to solve issues that prior application to process were not possible to predict.

D. Improvements

The activities performed in this phase are evaluation of current process in the software factory, development of a configuration management plan, implementation and validation. This will be followed by a process of continuous improvements. In short:

- Process implementation, identification of deviations and followed by continuous improvement
- Finally, analysis of proposed process impact.

This process has a requirements information update flow directly affecting the test suits and automated scripts as its differential. In other words, changes realized in software requirements will be able available during the whole

 ISBN: 978-1-61208-498-5

software development cycle, and consequently having an efficient management of automation tests.

The research approach presented here will result in a structured research process, which allows more control over activities and as scientific research process, will be applicable to new researches.

In the next session, we will present a proposal of integrated process to insert configuration management in automation tests, including phases, input and output artifacts, roles and responsibilities.

IV. PROPOSAL

The proposal presented here is partially a process of configuration management that might be implemented in a software development company in such way that it includes configurations items updates related to automatic tests. The proposed process objective is to maintain integrity, traceability of artifacts that affects automatic tests, and reuse of software components.

The process presented here was divided in two steps: The first step consists in evaluating the actual process of configuration management of a software factory; this will be used as study case resulting in the identification and analysis of gaps in the process. In the second step, we have a creation of integrated process of configuration management with the purpose to maintain integrity of artifacts during the whole software development cycle, developing the capacity to incorporate project scope changes. Bellow, we have a more detailed description of these two steps.

A. Actual configurations management process evaluation in a software factory

An analysis is performed to identify gaps related to items configuration control, and their impact in the utilization of automation tests tools; this step has two phases:

- Actual configuration management evaluation in a software factory chosen as study case. During this phase, we will use quality assurance such as process analysis and quality audits.
- Software development process evaluation. During this phase, we will use quality controls such as root cause diagrams, Pareto diagram, management of change revision and evaluation

B. Preparation of an integrated configuration management process

After the first step, we will create an integrated configuration management process to maintain integrity of the software requirements and automatic tests employed during the software development process. As part of the proposed process, we will define activities, roles, artifacts and tools, according description in Figure 2.

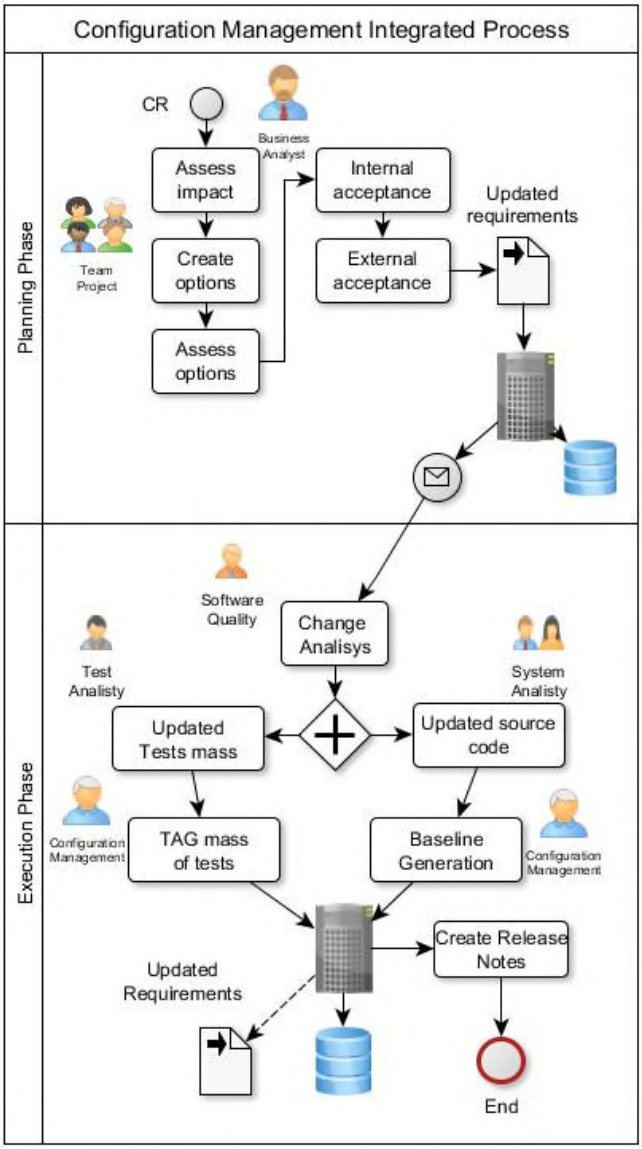

Figure 2. Integrated process flow chart. Source: Autor.

The integrated process is explained bellow

- The process is divided in between the two phases in a software development cycle, known as Planning and Execution Phase.
- During planning phase, we have as start point the changes request. The changes request might be initiated due to market reasons or regulation. The business analyst is responsible for the translation of the changes request into artifacts requirements. The change control is recorded in a bug tacker tool, making it possible to follow its evolution.
- Impact evaluation activities are performed by the project management that uses techniques such as learned lesson and expert opinion, as a technical evaluation of the changes request impact is needed. The changes request will be updated in a bug tracker tool.

 ISBN: 978-1-61208-498-5

- The options creation activity has as goal to generate many scenarios to fulfill the changes request, taking into consideration risk mitigation and impact into the product and finally project objectives. This activity is realized with the whole project team.
- Options validation has as purpose to simulate the implementation results obtained by the different proposed scenarios. After that, an unbiased selection process is possible, discarding the options that do not fit to the boundaries of project. The whole project team will realize this activity.
- After options validation, the next step is the internal acceptance, where consensus and finally approval and consensus of team on the solution to be applied to the changes request is obtained. The project management looks for consensus within project team.
- External acceptance is the final control activity in the integrated process; this activity has the purpose to obtain approval of project sponsor. The Project management is responsible to present and get approval from project sponsor.
- After the external acceptance of the integrated process for changes control is obtained, the software requirements will be updated. The System Analyst will be responsible for modification in software requirements. The approved change request is used as input for this activity, and as output, the software requirements updated should be available for the team in a repository. To finalize this activity, an automatic communication will be send to quality administration, reporting the requirements that were updated, and the modifications applied.
- The execution phase of project will be start when the communication on the software requirements update arrives.
- The software quality administration should analyze the changes, in order to guarantee the quality of the software development process. As responsible for this activity, we have a Quality Analyst. This activity has the updated software requirements as input. The analyst uses inspection and validation techniques. The updated software requirements keep a pendent status, as it reference sessions will have to wait for the new product version and new version to scripts designation to perform automatic tests.
- After Quality analysis, Systems Analysts modify the applications source code conform the updated software requirements. For this activity, we have the updated and QA verified software requirements as input. As output for this activity, we have a new software component for a new product baseline generation to be performed by the configuration management.
- In parallel to the changes implementation activity, the tests analysts will build the test suits and scripts for automatic tests. This activity has as input the updated software requirements, and previous versions test suits and scripts for automatic tests.
- The baseline generation activity, which has a configuration management as responsible, should update in the software requirements documents the product version that fulfills the requirements. , In addition, he should make the new package available as repository. In case of Release, it is necessary to send a package to distribution.
- The Tag creation activity for test suites and scripts used for automatic tests will be realized by Configuration Management, who should updated in software requirements document the artefacts version used for tests suites and scripts for automatic tests that attend this software requirements, in addition to verify, this item configuration in the repository.
- The Release notes creation activity, has as input the updated software requirements, test suites and scripts for automatic tests, product version, as well as components necessary to demonstrate it. The responsible to build this note is configuration management this activity finalizes the proposed process.

V. CONCLUSION AND FUTURE WORKS

This article presented a process proposal to support configuration management in automated software tests environments. The main objective was to provide better specific items configuration management between software requirements and automatic tests. Besides that, we aimed to improve artifacts traceability artifacts throughout software development cycle. The literature review has been finalized and we are working on the proposal implementation.

As this is a work in progress, we planned the proposal validation through a case study and survey, in order to evaluate the adherence and applicability this process. This will be combined with the identification of possible adjustment points in in view of its application to software development process.

The main obstacle in this proposal is the resistance against changes to the current software development process, and the absence of tools that support infrastructure of the proposed process. We used as constraints the minimization of the changes that impact the software development cycle inside the software factory used for case study. Other phases in the software development cycles are out of the scope of this research, e.g. project closure, business and software implementation processes.

The next step in this research is the implementation and evaluation of the proposed process in a software factory that builds payment applications and has a safe software development cycle.

REFERENCES

[1] M. Ferreira, C. Santos, T. Novais, and C. Albuquerque, “Gerência de Configuração para Testes Automatizados em uma Fábrica de

 ISBN: 978-1-61208-498-5

Software : Um estudo de caso Configuration Management to Tests Automations in a Software Factory : A case study.

[2] U. Ali and C. Kidd, “Barriers to effective configuration management application in a project context: An empirical investigation,” Int. J. Proj. Manag., vol. 32, no. 3, pp. 508–518, 2014.

[3] G. Antoniol, G. Canfora, G. Casazza, A. De Lucia, and E. Merlo, “Recovering traceability links between code and documentation,” IEEE Trans. Softw. Eng., vol. 28, no. 10, pp. 970–983, 2002.

[4] W. B. Frakes and S. Isoda, “Success factors of systematic reuse,” Software, IEEE, vol. 11, no. 5, pp. 14–19, 1994.

[5] O. C. Gotel, A. C. W. Finkelstein, and L. Sw, “An Analysis of the Requirements Traceability Problem Imperial College of Science , Technology & Medicine Department of Computing , 180 Queen ’ s Gate,” pp. 94–101, 1994.

[6] T. View, C. M. Plans, and T. View, “IEEE Standard for Software Configuration Management Plans,” IEEE Std, vol. 2005, no. August, pp. 0{_}1–19, 2005.

[7] K. Petersen and M. V. Mantyla, “Benefits and limitations of automated software testing: Systematic literature review and practitioner survey,” 2012 7th Int. Work. Autom. Softw. Test, pp. 36–42, 2012.

[8] O. C. Gotel, A. C. W. Finkelstein, and L. Sw, “An Analysis of the Requirements Traceability Problem Imperial College of Science , Technology & Medicine Department of Computing , 180 Queen ’ s Gate,” pp. 94–101, 1994.

[9] B. Kitchenham and S. Charters, “Guidelines for performing Systematic Literature Reviews in Software Engineering,” Engineering, vol. 2, p. 1051, 2007.

[10] P. Jamshidi, A. Ahmad, and C. Pahl, “Cloud Migration Research: A Systematic Review,” IEEE Trans. Cloud Comput., vol. 1, no. 2, pp. 142–157, 2013.

[11] F. Selleri Silva, F. S. F. Soares, A. L. Peres, I. M. De Azevedo, A. P. L. F. Vasconcelos, F. K. Kamei, and S. R. D. L. Meira, “Using CMMI together with agile software development: A systematic review,” Inf. Softw. Technol., vol. 58, pp. 20–43, 2015.

[12] P. Runeson and M. Höst, “Guidelines for conducting and reporting case study research in software engineering,” Empir. Softw. Eng., vol. 14, no. 2, pp. 131–164, 2009.

[13] A. Jedlitschka, M. Ciolkowski, and D. Pfahl, “Reporting experiments in software engineering,” Guid. to Adv. Empir. Softw. Eng., pp. 201–228, 2008.

[14] D. Tofan, M. Galster, P. Avgeriou, and D. Weyns, “Software engineering researchers’ attitudes on case studies and experiments: An exploratory survey,” Eval. Assess. Softw. Eng. (EASE 2011), 15th Annu. Conf., no. 638, pp. 91–95, 2011.

 ISBN: 978-1-61208-498-5