# FAST: Framework for Automating Software Testing

Ana Paula Furtado[1,2], Silvio Meira
[1]Informatics Centre – CIn
Federal University of Pernambuco
Recife, Brazil
e-mail: {apccf, srlm}@cin.ufpe.br

Carlos Santos, Tereza Novais, Marcelo Ferreira
[2]Recife Centre for Advanced Studies and Systems – CESAR
Recife, Brazil
e-mail: {carlosdombosco,terezanovais, marsantosfer}@gmail.com

***Abstract*—— The automation of software testing has played an important role in assessing the quality of software prior to delivering the product to the market. Several practices to introduce test automation are found both in the literature and in practice. However, most of these are not directly related to how automation practices could be systematically introduced into a software development context. Therefore, this paper describes a study which is still in progress on the best practices of test automation and how they can be systematically introduced into the software development process. It is in this context that this article presents and describes FAST – Framework for Automating Software Testing and does so by defining automation levels, areas and practice areas. The methodology used for this research is based on a systematic review of the literature, empirical research, a focus group and a case study. The initial general approach of the framework has been defined and will undergo this method of evaluation in order to collect feedback and identify improvements that need to be made in order to produce the complete version of the framework.**

***Keywords - software testing automation; software testing; process improvement; software quality.***

## I. INTRODUCTION

Test automation is the use of software to support test activities. It is considered an important topic of research and has been intensively studied in the literature [25]. However, despite its wide use, there are still gaps between existing approaches to test automation and its use in the software industry. The process of test automation needs time to mature: the creation of an infrastructure for tests for automation requires time and for automation-related processes to mature [25]. If the strategy for introducing automation in a project were to be inappropriate, this would not allow the company to reap the benefits related to test automation. Moreover, a large amount of time and resources is needed to support testing activities in the software development process [21]. For example, based on the model developed by Kit [17], it is estimated that software testing uses up to 80% of the total cost of software development, while the use of test automation could reduce the software development effort by up to 50%. Fewster [8] states that automating the running of tests is becoming more popular due to the need to improve the quality of software, whilst the complexity of software systems is becoming greater.

Despite the need to provide test automation techniques, there is still a lack of approaches and guidelines to assist with the design, implementation, and maintenance of test automation approaches [33]. Based on the gaps derived from observations in the software industry and on academic research, the problem related to this study can be stated as: **How should software test automation be introduced in the software development process?**

The main goal of this research is to produce a framework for software testing that could be used by the software industry to support the systematic introduction of test automation in the software development process. More specifically, we propose a test automation framework to reduce costs and improve product quality during the life-cycle of software development.

The rest of this paper is organized as follows. Section 2 gives a review of the literature followed, in Section 3, by a description of the research methodology. Section 4 introduces the technical approach and how it has been assessed so far. Finally, Section 5 contains some concluding remarks and offers suggestions for future studies.

## II. LITERATURE BACKGROUND

This study is based on the concepts and theory associated with testing in the software engineering domain, more precisely in the area of test automation. Many of the tasks and activities of tests can be automated, as can aspects of testing techniques. Many additional test tasks and activities can be supported by software-based tools, such as test case management; test monitoring and control; test data generation; static analysis; test case generation; test case execution; test environment implementation and maintenance; and session-based testing [14].

With a view to improving software quality, some studies present technical approaches to introduce testing within the software development context, by defining maturity models. Over the years, some maturity models and approaches have been developed, including models specifically related to the software testing area (those related to this study), as well as generic ones.

The first model to appear was the Software Capability Maturity Model – CMM-SW [23]. From that point on, some models appeared in order to present maturity models in the test process, such as MMAST [20], TAP [29], TMM [30], TCMM [2], TIM [7], TPI [19], TOM [32], TSM [11], TMMI [31], MPT.Br [10], and TAIM [6]. Besides these, some

 ISBN: 978-1-61208-498-5 

maturity models mention best practices for software development processes, without specifically focusing on testing discipline, such as CMM-SW [23], CMMI [28] and MPS.Br [27]. However, most of the test automation-related studies are defined as maturity models. These require levels of implementation and maturity assessment and this is one of the differences between these models and the one set out in this article.

In addition, except for Test Automation Improvement Model (TAIM), these models do not directly address techniques to introduce test automation into software development and therefore they do not answer the research question that this paper poses. On the other hand, TAIM presents a model based on measurement to support automation and the steps for improvement to be followed in 10 key areas and 1 general area. This approach is defined as a maturity model but it does not show what steps towards maturity must be taken in order to introduce automation.

Another approach related to this is the Maturity Model for Automated Software Testing (MMAST). This is a model that was developed for manufacturers of computerized medical equipment and its purpose is to define the appropriate level of automation into which an equipment manufacturer fits. It has four maturity levels: Level 1 - accidental automation, level 2 - beginning automation, level 3 - intentional automation and Level 4 - advanced automation. Despite being a maturity model, it has neither key areas nor process areas and its description is very broad and does not include matters as to how test automation can, in fact, be performed.

The Testing Assessment Program (TAP) is a maturity model which consists of 5 maturity levels, namely: initial and ad hoc (chaotic); repeatable and intuitive; defined qualitative; quantitative managed; and optimizing continuous improvement. Maturity is evaluated based on four key areas, namely: goals, people, management and techniques. however, the literature has only superficial descriptions of the model that impede it from making a more detailed analysis.

Test Maturity Model (TMM) is a model with 5 maturity levels: Level 1 - initial, Level 2 - phase definition, Level 3 - integration, Level 4 - management and measurement and Level 5 - optimization/ defect prevention and quality control. However, TMM does not discuss any issue directly related to test automation.

Testing Capability Maturity Model (TCMM) consists of 5 maturity levels: initial, repeatable, defined, managed and optimizing. The model includes key areas for each maturity level. However, the little information available does not describe TCCM appropriately so that what automation issues are present in the model can be analyzed.

Testability Support Model (TSM) was developed with a view to identifying actions that can improve the ability that a system has to be testable. This has three levels of maturity and 6 Key Support Areas, namely: Software Engineering Infrastructure, Project plans, Product information, Software design, Testware and Test environments. However, there is little information available on the model that enables it to be analyzed in greater depth.

Test Improvement Model (TIM) is a model intended to guide testing functions in their improvement work which has a four-step improvement ladder. The initial level has been given the number zero, as it is a non-compliance level and the other levels are numbered from 1 to 4, namely: Level 1 – optimizing, Level 2 – risk-lowering, Level 3 – cost-effectiveness and Level 4 – baseline. It has 5 key areas, namely: organization; planning and tracking; test case, testware and reviews. In its scope, testware deals with the actual testing procedures that are run, the support software, the data sets that are used to run the tests, and the supporting documentation. It includes managing the configuration of testware and the use of testware and tools. The model also mentions that tools can assist in performing non-creative and repetitive tasks, such as running the same test cases several times and automating testing activities. However, no guidelines are presented to support them

The Test Process Improvement Model (TPI) has 3 levels and 14 scales. Each level consists of a number of scales and these indicate which key areas need to be improved. The levels are: controlled, efficient and optimizing. The model also has 20 key areas, 1 of which is testware management. The model states that testing products (testware) should be maintainable and reusable and so they must be managed. Yet, test automation itself is absent in the model.

The objective of the Organization Testing Maturity Model (TOM) is to identify and prioritize organizational bottlenecks and generate solutions to these problems. A questionnaire is used to identify and prioritize both the symptoms and suggestions for improvement. Despite its name, it is not characterized as maturity model, as its focus is to solve problems and not improve testing in the organization, and there is no information on test automation.

The Test Maturity Model Integration (TMMi) is a model for improving the testing process developed as a guide and reference framework. It follows the staged version of CMMI, and also uses the concepts of maturity levels for evaluating and improving the testing process. TMMi consists of 5 maturity levels, namely: Level 1 - initial; Level 2 - managed; Level 3 - defined; Level 4 - measured; and Level 5 - optimization. Each level of maturity presents a set of process areas that must maturity at that level, in which each level of maturity is the starting point for the next level.

Despite being a maturity model specifically for the test area and its having systematic ways to enter the practice of software testing in the context of projects under development, it does not have a process area specifically dedicated to tools and/or test automation, nor does it include systematic suggestions for improving testing automation.

The Brazilian Maturity Model for Testing (MPT.BR) is a reference model that defines, implements and improves testing processes based on its being continuously improved. It also tackles the same approach to improving the testing process by using process areas that include the best practices of testing activities throughout the testing life cycle of the product. The model has 5 maturity levels, namely: Level 1 - partially managed; Level 2 - managed; Level 3 - defined; Level 4 - prevention of defects and Level 5 - automation and optimization.

Within the ambit of test automation, the model shows the process area of Automation of Executing the Test (AET), the

 ISBN: 978-1-61208-498-5 

purpose of which is to develop and maintain a strategy to automate the running of the test. This process area comprises the following list of specific practices:

- Defining the objectives of the automation regime;
- Defining criteria for selecting test cases for automation;
- Defining a framework for automating testing;
- Managing automated testing incidents;
- Ensuring adherence to the objectives of automation; and
- Analyzing the return on investment in automation.

Although the specific practices have a systematic way for introducing testing, they are still vague as to identifying the moment at which automation is to be performed. There is no specific information on introducing automation into the software development process, besides its not saying which testing levels can be automated. The written format is generic and comprehensive, into which every type of automation can fit. However, it does not help choosing where automation should start and what benefits can be achieved.

Therefore, this article puts forward a framework that can fill this gap in current research and aids taking a more flexible approach, for which there are no strict steps for introducing practices as this is in a maturity model. In this context, the next section will detail the research methodology associated with this study.

## III. RESEARCH METHODOLOGY

The research methodology planned for this study has three phases. The first is a **Bibliographical Review**, which comprises an exploratory review and a systematic review and the second is that of defining the **Proposal.** The latter is developed, underpinned by an empirical research including conducting interviews in the industry. The third phase is **Evaluation**, which will be conducted by using a focus group and a case study. Fig. 1 illustrates this approach.

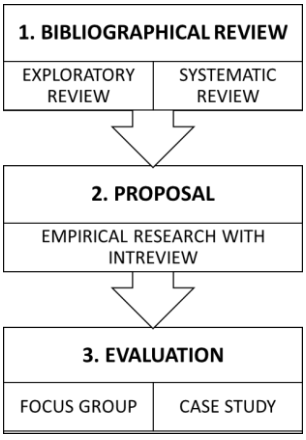


Figure 1. The design of the research activities | Source: author.

### A. Exploratory Review

An exploratory review, or a bibliographical review, is a critical, meticulous and comprehensive analysis of current publications in a given field of knowledge. It is an important step in the research, since it supports understanding the subject of research and assessment, if this is worth studying, and provides insights into how a researcher can define the scope for a particular area of interest [4]. The literature review correlates the research to the ongoing dialogue in the extensive literature, thereby filling in gaps and extending prior studies [22].

The main objective of this research instrument is to identify and explore publications related to the area being studied in order to learn how this problem has been approached and analyzed in previous studies with a view to reaching a better understanding of the research problem being investigated.

An ad-hoc bibliographical review has been undertaken by conducting searches of the scientific libraries available, such as IEEE Explorer, Engineering Village (including Inspec and Compendex), Scopus, ACM, Google Scholar and Springer.

### B. Systematic Review

According to Kitchenham [18], a systematic review of the literature is a way to identify, assess and interpret all relevant research available on a specific research question, or related phenomena of interest.

In order to achieve these benefits, a systematic review is under development which will be used to help assess the benefits and limitations of software testing automation and to analyze how the cost and quality of software is affected as a result of introducing automation practices.

In this research, an analysis is made of material published between 2005 and 2015, based on the main libraries such as IEEE Explorer, Engineering Village (including Inspec and Compendex), Scopus, ACM, Google Scholar and Springer. In addition, several relevant journals and conference proceedings are examined under the manual method. This is work-in-progress, during which data are being extracted by automatic searches. These data will be used to synthesize this study and report the results.

This systematic review is very important because it will ensure that all relevant studies in the literature are mapped. This will underpin how best to define the strategy needed to introduce test automation and guarantee that all related work is known and assessed in this research.

### C. Empirical Research with Interview

The empirical research with interview, based on experts' opinion, was one of the methods chosen to support this study. This consists of a comprehensive system for collecting information to describe, compare or explain knowledge, attitudes and behavior [24].

A group of experts in software testing automation was selected in order to collect their opinions, attitudes and expectations about the research questions for this study.

The survey was organized in three parts. The objective of **Part 1** was to gather personal information and information on the professional background. The goal of **Part 2** was to validate the problems of test automation, and this included analyzing the challenges, problems, benefits of the testing automation area and determining what gaps there are. **Part 3** focused on analyzing the automation strategy used in the

 ISBN: 978-1-61208-498-5 

companies, and included questions about the test strategy, levels of automation and technologies used. Finally, the aim of **Part 4** was to evaluate what opinions experts have as to the hypothesis of this study.

This survey was applied to 4 experts on testing, 2 of whom work in England and the other 2 in Brazil. They work directly on test activities in their companies and each of them has had more than ten years' experience in testing.

The results from **Part 2** mainly showed that there is a shortage of qualified professional who can engage on test automation and this makes it harder to introduce automation practices into a project. Moreover, the lack of senior management support also makes it more difficult to include automation practices in a project. Moreover, the difficulties faced in setting up an automation environment is also an impediment, as is the need for rework on tests assets due to changes made in requirements.

In **Part 2**, the benefits gained from test automation were: an increase in the team's velocity; more frequent delivery of working software; code continuous integration; fast execution of a group of test cases; parallel work can be done while tests are running; better visibility of code test coverage; and increasing the likelihood of finding new errors before delivering software to the market.

In **Part 3**, the intention was to collect experiences on how test automation was first introduced into a project. No results could be reached from this question, which re-emphasized the hypothesis of this study that there are no systematic ways to introduce test automation in a project that has not implemented this practice when that project was under development.

### *D. Focus Group*

Using a focus group is an approach in which a group of people gather to evaluate concepts and/or problems [3], and consists of a survey to obtain qualitative insights and feedback from experts in certain subjects. A focus group meeting involves semi-structured group interviews, in which the interactions in the group are explicitly used to generate data. Participants offer personal opinions but can also interact based on the response of other participants while the interviewer acts as moderator so that the interview remains focused on its objectives [9].

The objective of the focus group in the context of this paper is to evaluate the proposal of this thesis with a view to collecting suggestions for improving and developing the proposal prior to conducting the case study.

### *E. Case Study*

A case study can be defined as a research strategy on understanding the dynamics present in a given environment, in accordance with the view of Eisenhardt [5]. A case study is an empirical method that targets analyzing a phenomenon in a given context. The purpose of the case study is to seek pieces of formal evidence by using variables that can be measured and to draw inferences coming from the example for a given population.

Case studies are appropriate when the boundaries between the phenomenon and the context are not clearly defined, and the type of evidence is considered to be very rich and contextualized [9].

In this context, a case study should be used as a tool to validate the solution proposed, and will be conducted as proposed by Runeson and Höst [26], based on the following steps:

- Designing the case study;
- Preparing for data collection;
- Collecting evidence;
- Analyzing the data collected; and
- Writing a Report.

The case to be applied will be in a software development company that has an academic management product that integrates all areas of the educational institution. The data collection method will start from the principle that the researcher will have direct contact with the data and collect them in real-time (first degree data), by using interviews and focus groups.

Data analysis shall be conducted quantitatively, using correlation analysis, which describes how a given measurement of a process activity is related to the same measurement in a previous process, and thus compare them. The measurement being compared is the cost of testing, by assessing whether it decreases when automated testing is introduced into the software development process. In addition, the quality of the software shall also be analyzed from when automated testing was introduced in the software development process.

Based on all observations so far gathered, in line with the steps defined in the research methodology, the technical approach has been developed and will be detailed in the following Section.

## IV. TECHNICAL APPROACH

The approach developed to support the objective of this study and to answer the research questions is the **Framework for Automating Software Testing (FAST)**.

According to ISO/IEC/IEEE [16], a framework can be defined as "a reusable design (models and/or code) that can be refined (specialized) and extended to provide some portion of the overall functionality of many applications".

Although conceptually, the term 'framework' is more related to the technical component of a software, this study uses this term in order to make it clearer how a group of best practices can be adapted to a project in accordance with its specific needs so as to reap the best benefits of software testing automation. FAST differs from a maturity model in that the practice areas are not mandatory and there is no need to certify a company in the framework; and in accordance with the needs of a specific environment, each process area can be applied to a project.

Therefore, FAST is defined in accordance with the components shown in Fig. 2 and described as follows:

- **Automation level.** Determining this is a separate test effort that has its own documentation and resources [15]. This represents the scope within which automation activities will be welcomed in a project;

- **Area** is a general range of interest in which FAST is divided into two parts, Technical and Support, as shown in Fig. 3. It includes what is needed to introduce testing automation techniques into a project and consists of process areas;
- **Process Area** This is a group of related practices in an area that, when implemented collectively, satisfies a set of goals considered important for enhancing that area [28]. Each process area is assigned a specific **purpose**, has **guidelines** that must be implemented, and suggested **work products** that must be produced by engaging on such practices.

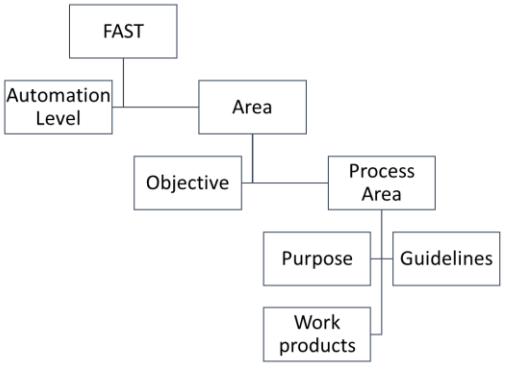


Figure 2. FAST components | Source: author.

The relationship between the areas were defined in accordance with CMMI-DEV [28], where the support process areas address processes that are used in the context of performing other processes. In this case, the Support Area comprises a fundamental support function and relies on the processes of the Technical Area for input. For example, the process area for Project Planning will plan the test strategy for the Process Area of Unit Testing.

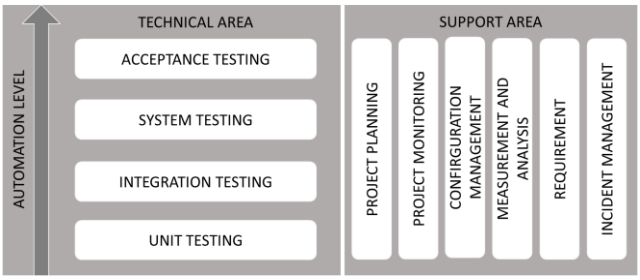


Figure 3. FAST areas | Source: author.

Fig. 3 presents the overall structure of FAST, together with the process areas for each area. The framework can be applied by instantiating it in a project context, where the process areas can be adapted to best fit the environment where it will be applied. The objectives and process areas will be described in the following section.

### *A. FAST Support Area*

The objective of the FAST Support area is to cover essential mechanisms to support establishing and maintaining the automation environment. It was developed based on the reference model of both CMMI-DEV, which covers a generic view of best practices for software development projects, and TMMI, which has a group of guidelines specific to test projects. The objective of process is to undertake practices that are fundamental to systematically introducing automation practices but are not specifically directed towards automation practices. The objective and guidelines of the process areas from the support area are given below.

***Project Planning***

The purpose of Project Planning is to define a plan to support setting up an automated test project for which the **guidelines** are as follows:

- Plan test project;
- Define test strategy;
- Make estimates;
- Analyze project and product risks; and
- Obtain commitment to the plan.

The **work products** related to this process area can be a test plan which has the information required by the guidelines.

***Project monitoring***

The purpose of Project Monitoring is to provide an understanding of the project's progress so that appropriate corrective actions can be taken when the project's performance significantly diverges from the plan [28].

It has the following **guidelines**:

- Monitor progress against the plan;
- Monitor product quality;
- Conduct corrective actions as per the demand; and
- Manage corrective actions to closure.

The **work products** related to this process area can be defined as project monitoring sheets, together with systems to track adherence to the project schedule and issues from start to closure.

***Configuration management***

One reason maintainability is so important is that without it tests cannot be accumulated. Therefore, the purpose of Configuration Management is to establish and maintain the integrity of testware by defining the management system for the configuration, for which there are the following **guidelines**:

Establish project baselines;

- Control and track changes; and
- Establish the integrity of the project.

The main **work products** related to this process area are the configuration management system together with its plan.

***Measurement and analysis***

The purpose of this process area is to define, collect, analyze and apply measurements to support an organization in objectively evaluating the effectiveness and efficiency of the test process [31]. In this case, all indicators defined are specifically aligned to the automation strategy, in order to best achieve its desired objectives. The **guidelines** for this are as follows:

- Define test indicators for the project;
- Specify test measures in terms of data collection and storage procedures;

- Specify analysis procedures;
- Collect test measurement data;
- Analyze test measurement data;
- Communicate results; and
- Store data and results.

The main **work product** here is the measurement and analysis plan, together with the data collected and formally analyzed.

***Requirement***

The purpose of this process area is to clearly define the test automation requirements and the following **guidelines** area associated with this process area:

- Define what products need to be automated;
- Prioritize requirements; and
- Maintain the traceability of requirements.

The **work products** in this case are those on the list of requirements, for which tests will be automated in the software development process.

***Incident management***

The purpose of this is process area is to objectively define the mechanism and procedures to formally monitor all product incidents derived from test automation activities. It supports testers in the investigation and documentation of test incident reports and the retesting of defects when required [10].

In order to achieve this, the following **guidelines** are suggested:

- Establish incident management system;
- Register, classify and prioritize incidents;
- Solve and track incident upon its closure; and
- Escalate non-solved incidents.

The main **work product** related to this process area is the incident management system.

*B. FAST Technical Area*

The objective of the FAST Technical area is to establish and maintain automated mechanisms for testing software applications throughout test levels in order to produce test environments that can be developed, managed and maintained efficiently. The technical area consists of four process areas, in accordance with the automation levels that a software can undergo, and its objective is to describe practices to support automation activity. The details of the processes areas of the technical area re given below.

***Unit Testing***

The objective of this process area is to provide mechanisms so that unit tests are implemented in a systematic and documented way in order to maximize the benefits of automation in the test project. This area has the following **guidelines**:

- Design the test suite;
- Implement the refined plan and test design;
- Measure the test unit;
- Run the test procedures;
- Evaluate test completion; and
- Evaluate the effort and test unit.

The **work products** related to this process area are the test items, the design of the test specification [15], test summary report [15], and reporting the failures found.

***Integration Testing***

The purpose of the Integration Testing process area is to evaluate the integration between software components, to ensure that the architectural design of the system is implemented correctly [13][13]. It tests the interface between components and interactions in different parts of the system, such as operating system, the file system and the interface between the systems [1]

This process area has the following **guidelines**:

- Design the integration approach (bottom-up or top-down), in accordance with the system's requirements;
- Design the set of integration tests;
- Implement the design of the integration tests:
- Run the test procedures;
- Assess whether the tests have been completed and whether they have achieved the required coverage of the requirements; and
- Formally record and direct the non-conformities and restrictions arising from the integration actions.

The **work products** related to this process area are the set of integration tests and formally reported results.

***System Testing***

The purpose of the System Testing process is to test the integrated and complete systems so as to assess its ability to communicate with each other and validate whether the systems are in accordance with the specifications of the requirements [15].

The objective of this process area is to test the finalized system and analyze the behavior of the system as a whole in order to analyze compatibility with the specified requirements, and can be performed in line with the testing approach selected, such as risk-based products, business processes or other description of the behavior of a high-level system [1].

This process area has the following **guidelines**:

- Establish the testing approach of the system;
- Select the testing techniques of the system;
- Design the set of system tests;
- Implement the system tests;
- Run the system tests;
- Assess whether the test was completed; and
- Register, formally, and direct non-conformities.

This process area needs to address questions about the selection of requirements so as to generate a group of test cases to be automated and run in the context of a project. The related **work products** are the set of test cases, and the formal record in a specific tool of the results.

***Acceptance Testing***

The objective of the Acceptance Testing process area is to ensure that the product is working and that it can be presented for acceptance, in which the customer and/or user is expected to be involved [12]. At this level of automation, the object to be tested is the complete system and this must address

 ISBN: 978-1-61208-498-5 

activities in order to demonstrate the customer's acceptance regarding the final system.

The objective of this process area is to ensure that the suite of acceptance tests, planned in accordance with the strategy and needs of the end user, is implemented so that it can run automatically. To this end, this process area has the following **guidelines**:

- Define acceptance criteria;
- Define the acceptance plan;
- Prepare the testing acceptance environment;
- Assess the conditions of acceptance; and
- Conduct the closure of acceptance.

The **work products** related to this process area are the acceptance plan, the acceptance environment, the suite of acceptance tests, the due register of the test results and incidents recorded and followed upon until closure on the appropriate tool.

In this context, this section presented the FAST way to include the theoretical structure and its process areas. The following section presents the conclusions and future studies planned for this research.

## V. CONCLUSIONS

This paper proposes a framework to support the systematic introduction of test automation in the context of software development. The approach was defined by describing automation levels, technical and support areas, and practice areas.

In order to evaluate this general approach, a plan was drawn to conduct a focus group and a case study, in accordance with the descriptions given in the methodology Section, in order to gather feedback on the value of FAST being feasible, complete and adequate. After this phase, it was expected that the description of the framework would be enhanced in order to finalize how to define the framework.

Some threats to this study were identified and are being dealt with in order to minimize side effects to the expected results. The first threat is the possibility of bias while analyzing data from the case studies, since the results of introducing FAST may vary according to the domain of application. In order to minimize this threat, three different scenarios and domains were planned to be part of the scope of the case study, in which the absence of information in a specific context can be complemented by having it in another.

Another threat would be to focus the definition of the framework upon the perspective of the very few authors found in the literature review. Hence, to diminish this possible problem, a systematic review of the literature is being developed to guarantee that all research studies are taken into consideration when developing this project.

Therefore, this work-in-progress offers contributions to research on test automation and its practice, whereby a framework is compiled from a combination of experience, practice and a systematic review of the literature in the form of best practices, which can be applied when running tests in software development.

## ACKNOWLEDGMENT

This research work was supported by the Brazilian National Research Council (CNPq) of the Ministry of Science, Technology and Innovation of Brazil, process #206328/2014-1. The international cooperation with the Open University was part of the Science without Borders program (http://www.cienciasemfronteiras.gov.br/web/csf).

## REFERENCES

[1] R. Black, E. Veenendaal and D. Graham, Foundations of Software Testing ISTQB Certification. 3rd Edition, Reino Unido, January 2012.

[2] S. M. Burgess and R. D. Drabick, "The I.T.B.G.testing capability maturity model (TCMM)", 1996, available from https://wwwbruegge.informatik.tu-muenchen.de/lehrstuhl_1/files/teaching/ws0708/ManagementSoftwareTesting/12-4-1-FPdef.pdf, retrieved: July, 2016.

[3] S. Caplan, "Using focus group methodology for ergonomic design. Ergonomics", v. 33, n. 5, p. 527-33, 1990.

[4] J. Creswell, Research design: qualitative, quantitative and mixed methods approaches, 4th Edition, Washington D.C., 2014.

[5] K, Eisenhardt, Building theories from case study research. The Academy of Management Review, October 1989.

[6] S. Eldh, K. Andersson, A. Ermedahl and K. Wiklund, "Towards a test automation improvement model (TAIM)", in 014 IEEE Seventh International Conference on Software Testing, Verification and Validation Workshops (ICSTW), Cleveland, pp. 337-342, April 2014.

[7] T. Ericson, A. Subotic and S. Ursing, "TIM - A test improvement", Software Testing Verification and Reliability 7(4), pp. 229-246, December 1998.

[8] M. Fewster, Common Mistakes in Test Automation, Grove Consultant, 2011.

[9] S. Fincher and M. Petre, Computer Science Education Research. Taylor and Francis, January 2004.

[10] A. P. Furtado, M. Gomes, E. Andrade and I. Farias, "MPT.BR: A Brazilian maturity model for testing", in The 12th International Conference on Quality Software (QSIC), Xi'an, pp. 220-229, August 2012.

[11] D. Gelperin, "A testability support model (TSM)", in the fifth International Conference On Software Testing, Analysis & Review, Orlando, Florida, pp. 13-17, May 1996.

[12] A. M. Hass, Guide To Advanced Software Testing, Artech House, London, 2008.

[13] IEEE 610.12 Standard Glossary of Software Engineering Terminology, IEEE Computer Society, 1990.

[14] IEEE 29119-1: Software and Systems Engineering – Software Testing – part 1: Concepts and Definitions, IEEE Computer Society, 2013.

[15] IEEE 829 Standard for Software and System Test Documentation, IEEE Computer Society, 2008.

[16] ISO/IEC/IEEE 24765 International Standard, Systems and Software Engineering Vocabulary, 2010.

[17] E. Kit and S. Finzi, Software testing in the real world: improving the process, Addison-Wesley Publishing Co., New York, 1995.

[18] B. Kitchenham and S. Charters, Guidelines for Performing Systematic Literature Reviews in Software Engineering, Software Engineering Group, School of Computer Science and Mathematics, Keele University, Tech. Rep. EBSE-2007-01, July 2007.

[19] T. Koomen and M. Pol, Test Process Improvement (TPI), A Practical Step-by-step Guide to Structured Testing, Addison-Wesley, 1999.

[20] M. E. Krause, "A maturity model for automated software testing", in Medical Device & Diagnostic Industry Magazine, December 1994, available from https://wwwbruegge.informatik.tu-muenchen.de/lehrstuhl_1/files/teaching/ws0708/ManagementSoftwareTesting/12-4-1-FPdef.pdf, retrieved: July, 2016.

 ISBN: 978-1-61208-498-5 

[21] H. K. Leung and L. White, "A cost model to compare regression test strategies," in 1991 Conference on Software Maintenance, Sorrento, pp. 201–208, October 1991.

[22] C. Marshall and G. Rossman, Designing Qualitative Research, SAGE Publications, Washington, DC, USA, 2011.

[23] M. Paulk, C. Weber, S. Garcia, M. Chrissis and B. Bush, "The Capability Maturity", version 1.1., IEEE Software 10, no3 pp. 18-27, 1993.

[24] S. L. Pfleeger and B. A. Kitchenham, "Principles of survey research: part 1: turning lemons into lemonade", ACM SIGSOFT Software Engineering Notes, 26(6), pp. 16-18, 2001.

[25] D. M. Rafi, K. R. K. Moses, K. Petersen and M. V. Mäntylä, "Benefits and limitations of automated software testing: systematic literature review and practitioner survey", in 2012 7th International Workshop on Automation of Software Test (AST), Zurich, p.p. 36-42, June 2012.

[26] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering", In: Empirical Software Engineering Journal, Volume 14, Issue 2, pp. 131-164, April 2009.

[27] Softex, "MPS.BR – Brazilian software process improvement", General Guide, V1.2., Rio de Janeiro, Softex, 2007.

[28] Software Engineering Institute (SEI). CMMI for Software Development, version 1.3, staged representation, Pittsburgh, PA, 2010. Available from: www.sei.cmu.edu/reports/10tr033.pdf, CMU/SEI-2010-TR-033, retrieved: July, 2016.

[29] TAP, Testing Assessment Program (TAP), Software Futures Ltd and IE Testing Consultancy LTD, 1995.

[30] TMM, Test Maturity Model, Illinois Institute of Technology. , Available from http://science.iit.edu/computer-science/research/testing-maturity-model-tmm 2014.08.11 retrieved: July 2016.

[31] TMMI, Test Maturity Model Integration, Release 1.0, TMMi Foundation, Ireland, 2012, Available from http://www.tmmi.org/pdf/TMMi.Framework.pdf 2014.08.11, retrieved: July 2016.

[32] TOM apud R. Swinkels, A Comparison of TMM and Other Test Process Improvement Models, Project Report 12-3-1-FP, 2000.

[33] K. Wiklund, S. Eldh, D. Sundmark and K. Lundqvist , "Technical debt in test automation", in 2012 IEEE Fith International Conference on Software Testing, Verification and Validation (ICST), Montreal, pp. 887-892, April 2012.

 ISBN: 978-1-61208-498-5 