1. Introduction

One of the cores of the software testing process is the implementation and testing execution phase. This phase includes, among other activities, specifying the procedure to be adopted in addition to all the information necessary for execution of the tests,for example, the environment preparation and device configuration [1].One of the key points of this phase, is the configuration activity of the devices to be tested and loading data necessary for testing execution, this activity consist in the selection and inclusion of this dataand, carrying out necessary settings to run tests. Usually this activity is performed manually by testers, following the procedures listed during the analysis and modeling of test cases [2]. Mistakes made in the preparation procedures may lead to inaccurate results, in addition to increasing the consumption of time required in this activity.

In recent years, the automation of software testing has been widely adopted by technology companies.Due, the possibility of repetition of the certain testing activities, with low cost combined with an increase in test coverage [3]. Test automation is seen as an alternative for improvement and efficiency of the testing process as well as, a means to ensure the quality of a product under development [2].

The automation of testing procedures, can also be adopted to ensure a correct configuration of the item being tested. The automate the configuration procedures is useful when it is necessary to have different settings on multiple devices or even to unify this setting.

The Project CIn / Motorola, a partnership of the Center for Informatics of the Federal University of Pernambuco, with Motorola mobility,develops testing activities on mobile devices.Tests conducted on the project using various approaches: some tests are performed manually, automated or semi-automated manner.With respect to automated testing, the creation and use of test scripts, is one of the means adopted for the implementation of this approach in the Project CIn / Motorola environment.This scripts are responsible for the configuration and, in some cases they also perform the test.

In most cases, the scripts are created to perform interaction with GUI elements, simulating manual use of the device.However, this technique has some problems, according to [4] automated tests has a Strong dependence with GUI components of the application. Changes in graphical user interface of the device such as: a different kind of screen resolution and addition of a new prompt for example, may hinder the Script reuse for different devices.

Moreover, another problem that often occurs in the environment of the Project CIn/Motorola is to changes in the system versions (builds), as well as the region where this system will be adopted. This entails a change in the position of elements in the GUI and even adding new screens, generating inaccuracy in executing automated tests. Another issue that arises in this approach is portability, as some scripts are written to be compatible with the Linux system, while there are scripts that run only under Windows.

To run tests on different platforms, the script can not be reused, it is necessary for it to be translated to the platform which tests will be performed. According to [4], one of the ways to solve this problem is through the creation of flexible scripts to allow, if necessary a quickly adapt to the changes occurred during the execution of the test. Techniques such as Keyword-driven and data-driven, has been applied with the aim of making the Scripts more maintainable and reusable (Fewster, 1999).

The solution adopted by the Project CIn / Motorola to resolve this situation, is to dedicate a member of the test team and part of time to running the tests for the adaptation of the script. This solution, adopted to try ensure that the script is in accordance with the changes caused by the different builds, and if applicable, migrate the Script to another platform where testing should be performed.

Therefore, this work presents the development of a prototype for testing setup automation. Given the need for greater accuracy in testing configuration procedures. Regarding test scripts, the develop of this prototype, aims reduce the need to translate the scripts, through techniques such as Keyword-driven and use of technologies, as Python, Java and Android. The technologies adopted for the development of the prototype, allow the adoption of the tool by the various platforms used in the Project CIn / Motorola environment, and provides an architecture that enables greater granularity and better match the frequent changes in the builds of the devices under test.

The development of this tool was based on a survey conducted with collaborators of the Project CIn/Motorola, which sought to observe what kind of test, a wrong configuration would cause greater damage, inaccuracy or consumption of time. The data showed the CTS (Confirmation …), as the test that small mistakes in the test setup would cause the loss of an entire execution, since the device is configured the test running lasts about six (6) hours.

Initially, this proposal covers only the configuration of CTS tests. The initial modules of the prototype, were implemented in a flexible architecture that allows the addition of new modules for the future other types of tests are aggregated to this solution. Only the GUI, need to be reformulated for later versions, because it has been designed only for the initial needs of the development of the tool.

The contributions of this work for the environment of the Project CIn / Motorola include;

* Increase the ability to run tests: Reducing time spent in device configuration;
* Increase the precision in the results;
* Avoid rework: Through a multi-platform solution no will be necessary to update or rewrite the test scripts

This paper is organized as follows, Section 2 contains the theoretical review and a survey of the techniques used in this work. Section 3 describes the development of the prototype, showing the phases of the process, how was the survey of the development of the tool, relevant data as well as all the diagramming the use cases and a description of the project manage. Section 4 presents the results obtained in this work, Section 5 provides the conclusion and final remarks.

2. Theoretical Referential

2.1 Software Testing

According to Pressman (2006), software testing can be described as a set of activities for the verification and validation of software, these can be planned in advance and conducted systematically. To Sommervile (2007), the test is a dynamic technique for verification and validation, which involves running a program with a set of input data and verify that conforms to the expected result. However verify and validate have different meanings, the verification attests if the application is in according to their specification, in turn, validation verify if the results is in according to expected result by the clients.

To Pezze and Young, 2008, to evaluate the degree to which a software system really meets your needs, in order to meet the actual needs of the user, is called validation.

Validation against real requirements, necessarily involves human judgment and a potential for ambiguity, misunderstanding or disagreement. The specification must be sufficiently precise and unambiguous to the point that there can be no disagreement about the acceptability of a given system behavior (Pezze and Young, 2008). Atender requisitos não é o mesmo que estar em conformidade com sua especificação. Entende se como especificação uma declaração sobre uma solução particular apresentada por um problema, sendo a solução proposta pode ou não atingir seu objetivo.

Para Delamaro et. al. 2007, as atividades de verificação e validação devem ser conduzidas durante todo o processo de desenvolvimento de software, partindo de sua concepção, sendo estas divididas em estáticas ou dinâmicas. As estáticas não requerem a execução ou mesmo a existência de um aplicativo ou modelo executável para ser conduzido porem as dinâmicas são baseadas na execução de uma aplicação ou modelo. Revisão, inspeção, técnicas de leitura dentre outras, são considerados atividades estáticas de verificação e validação. A avaliação dinâmica, foca avaliar o comportamento do software em tempo de execução, objetivando provar não conformidades nos resultados apresentados.

A literatura apresenta algumas definições e termos utilizados em testes de software, de acordo com Delemaro et. Al. 2007, são :

Fault: Processos ou definições de dados incorretos;

Mistake: Ação humana que produz um resultado incorreto;

Error: Durante a execução de um programa é caracterizado por um estado inconsistente ou inesperado;

Failure: Resultado produzido é diferente do resultado esperado.

O padrão IEEE (IEEE STD. Glossary of Software Engineering Terminology, padrão 610.12/1990) define que os problemas introduzidos no software pelo desenvolvedor são chamados de falhas (fault). Enganos podem ser cometidos tanto na especificação quanto no código do sistema. Quando uma falha é ativada durante a execução do software, um erro é gerado (error). Caso o problema se manifeste nas fronteiras do sistema, ocorre um defeito (failure), que pode ser percebido pelo usuário.

Conforme Pressman (2006), a presença de defeitos indica falhas no software, consideram se casos de testes bem sucedidos, os que revelam falhas que não tenham sido descoberta