TaRGeT Development Process Specification

Last Update: 14-Oct-2009

ABSTRACT: TaRGeT is a tool designed to automatically generate test from suitable input use case documents. This document describes the requirements of this tool.

KEYWORDS: Use Cases, Requirements, Automatic Test Generation

Revision History

Revision #	Date	Author	Description
01.00-D01	14-Oct-2009	Laís Neves(lmn3)	Initial draft.
01.00-D02	4-Nov-2009	Michele Silva (mcms)	Revision.

Contents

1. Introduction

This document has the purpose to describe the development process for TaRGeT tool.

This document set the bounds for each person's roles and responsibilities in the TaRGeT development process so that the collaboration is successful and efficient.

1.1 Abbreviations, Acronyms and Definitions

TaRGeT	Test and Requirements Generation Tool
SPL	Software Product Line
CR	Change Request
MIT	Massachusetts Institute of Technology

1.2 References

[1] TaRGeT Architecture Document

[2] TaRGeT Requirements Document

[3] TaRGeT Use Case Document

[4] SVN Book - Using Branches (http://svnbook.red-bean.com/en/1.0/ch04s02.html)

[5] Code Conventions for the Java Programming Language (http://java.sun.com/docs/codeconv/html/CodeConvTOC.doc.html)

[6] Eclipse IDE (http://www.eclipse.org/)

[7] A Framework for Software Product Line Practice. Software (http://www.sei.cmu.edu/productlines/frame_report/process_def.htm)

[8] Michalis Anastasopoulos. Software Product Lines Evolution. RiSE Summer School 2008.

[9] MIT License (http://pt.wikipedia.org/wiki/Licença_MIT)

[10] Bug Reporting Guidelines http://www.chiark.greenend.org.uk/~sgtatham/bugs.html

[11] License address at SVN

1.3 Overview

TaRGeT is a working environment that is capable to visualize and navigate through input use case documents in order to generate test suites. The inputs and outputs of the tool are organized as a project.

TaRGeT is organized as a software product line (SPL) that integrates different features of TaRGeT products.

This document is organized as follow: section 2 describes the principles of TaRGeT development process, section 3 describes the structure and organization of different roles involved in the process, section 4 describes how the process applies for code artifacts, section 5 describes how the process applies to other development artifacts, section 6 describes the process related to formal and informal releases and finally section 7 describes the TaRGeT development license.

1.4 Problem Reporting Instructions

Problems and corrections to this document should be reported to Érica Hori (eaah@cin.ufpe.br), Laís Neves (lmn3@cin.ufpe.br), Michelle Silva (mcms@cin.ufpe.br) or Paulo Borba (phmb@cin.ufpe.br).

2. Principles

All TaRGeT projects should strive for the highest possible quality in the delivered products.

The development process should guarantee that all core assets from the TaRGeT software product line will be consistent during all the development phases.

3. Structure and Organization

This section describes the main roles that are involved on TaRGeT development process.

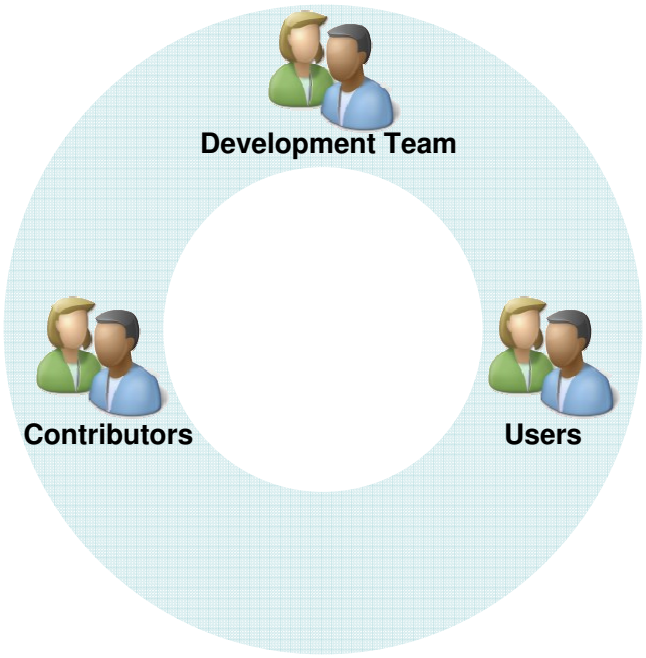

Fig 3.1 – Main Roles

Development Team: TaRGeT has a permanent development team located at CIn/Motorola Research Project.

This team is responsible to maintain the TaRGeT project (including code and documentation artifacts), to develop the requests performed by Motorola´s users and to interact and provide support to contributors. It is also responsible to discuss, approve or reject the modifications on the TaRGeT SPL architecture in order to assure the quality and modularity of all TaRGet products.

The main plugins that compose the TarGeT SPL architecture are listed below:

- Core
- Common
- Project Manager
- Test Case Generation
- Input and Output plugins
- Consistecy Management

For more information about TaRGeT plugins and architecture please read the **TaRGeT Architecture Document** [1].

Contributors: TaRGet is also an extensible platform for contributors develop their own functionalities according to their specific needs. Currently TaRGeT already supports some features variations, such as:

- Input and output formats
- Use Case Edition
- Test generation
- Filters

The rules and guidelines for the development of new contributions will be listed in the next sections of this document.

Users: This group includes all users of TaRGeT tool. They may use different version of TaRGeT products according to their specific needs.

4. Code

This section describes the process related to the development of code artifacts to the TaRGeT Software Product Line.

4.1 Development Environment

The development team uses the Eclipse IDE [6] to develop TaRGeT features. The used distribution is Eclipse for RCP/Plug-in Developers. In addition, the following plug-ins should also be installed:

- Eclipse AspectJ Development Tools – used to support development with AspectJ applications
- Subclipse – used to version control
- Nebula Grid Plug-in – used to some graphic interface components

Developers shall use the TaRGeT specific code template and formatter files that are located in the project's SVN repository.

TaRGeT code follows the guidelines defined in the Java code conventions document [5].

4.2 Repository Organization

All core assets of the TaRGeT SPL are located in the project SVN repository at CIn/UFPE. The repository folders are organized as below:

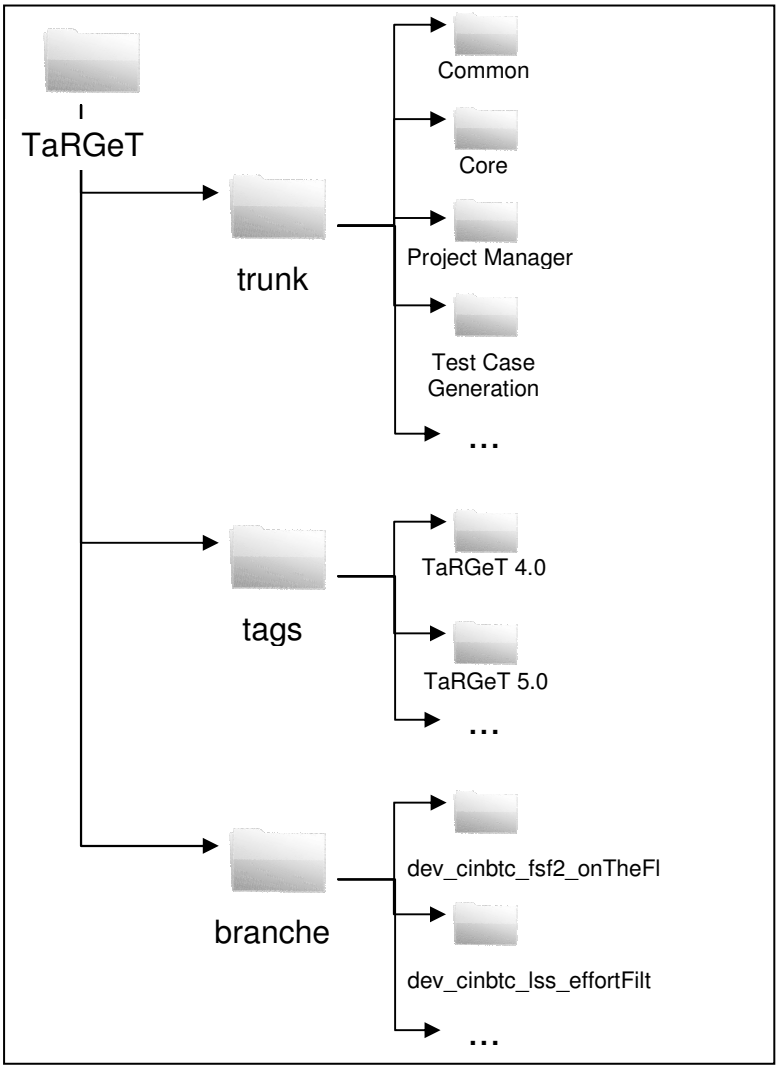

Fig 4.1 – TaRGeT Repository Organization

- **Trunk:** Contains the last version of code that is under development by the team

- **Tags:** Contains stable versions of released code.

- **Branches:** Contains the different branches of TaRGeT projects. All contributors' work will be placed in branches. Branches names follows this name convention:

  dev_siteName_developerLogin_smallDescription

  where:

- Site name: the name of the entity where the contributor is located. The site name for the development team will be **cinResearchProject**

- Developer login: the login of the developer that will use the branch

- Small Description: a small description about the functionality that will be developed in the branch

The branching name conventions will be applied only to the new branches created in the repository. The existing branches shall remain with the current name.

For more information about branching see [4].

4.3 CIn Development Team Responsabilities

One of the responsibilities of the development team is to manage the organization of TaRGeT SVN repository. The activities include the creation of project's branches and maintenance of project trunk.

The team is also responsible for develop the change request performed by Motorola's users, manage the changes on TaRGeT architecture and maintain the core assets of TaRGeT SPL.

The development team should also provide support for contributor's interactions.

4.4 Contributors Responsabilities

Contributor's development will be located in branches of TaRGeT trunk on the project's SVN repository. These branches should be created by the development team and they will follow the naming conventions defined in subsection 4.2. The picture below shows a contributors branch and the development new TaRGeT projects.

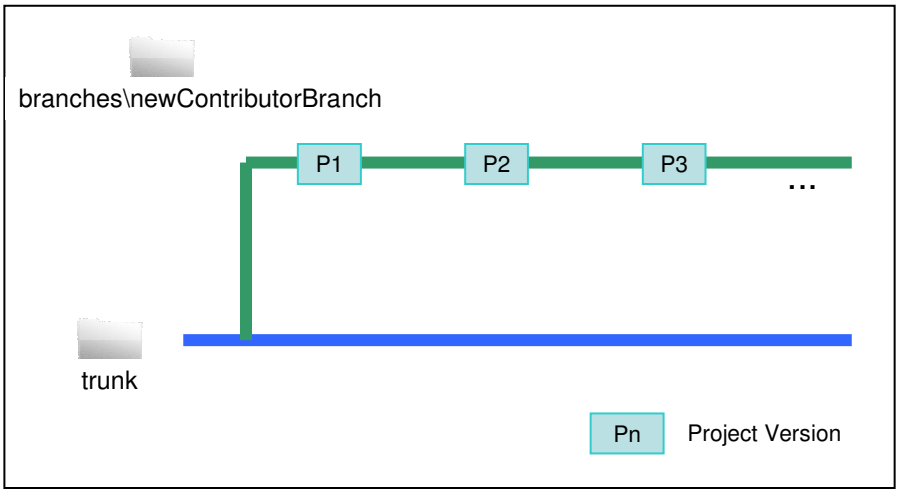

Fig. 4.2 – Contributor's development branch

The TaRGeT Development Process identifies three scenarios for contributor's activities:

- Development of a new variation
- Modifications on TaRGeT Architecture
- Bug correction

Each scenario will be explained in the following subsections where the roles and activities of contributors and of the development team will be detailed.

4.4.1 Development of a new variation

This scenario describes the situation when a contributor wants to create a new variation for an existing variation point, like new input or output plug-ins. TaRGeT architecture already provides some extension points as listed in the figure below:

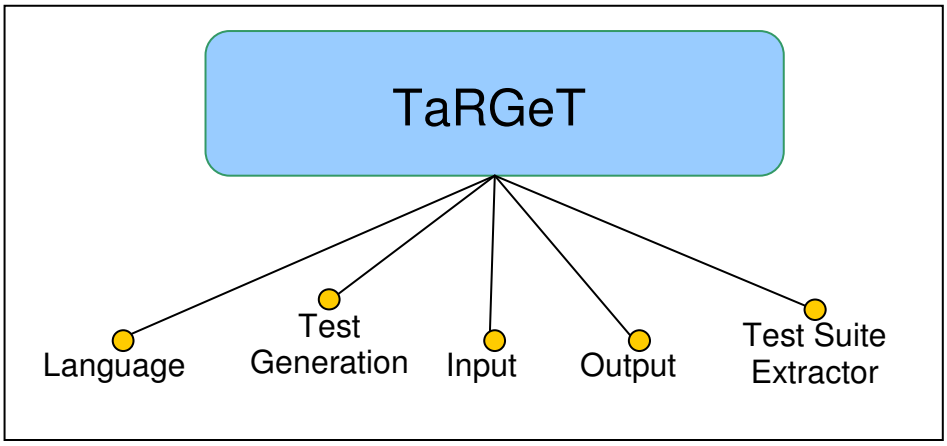

Fig 4.3 – Main TaRGeT exptension points

In most cases, implementations of new variations are made in independent plug-ins projects. This is a benefit extended from the Eclipse RCP architecture. Assuming that these new projects do not affect other TaRGeT SPL projects and that they obey the development guidelines described above, they can be merged directly to the SVN trunk as illustrates the picture below.

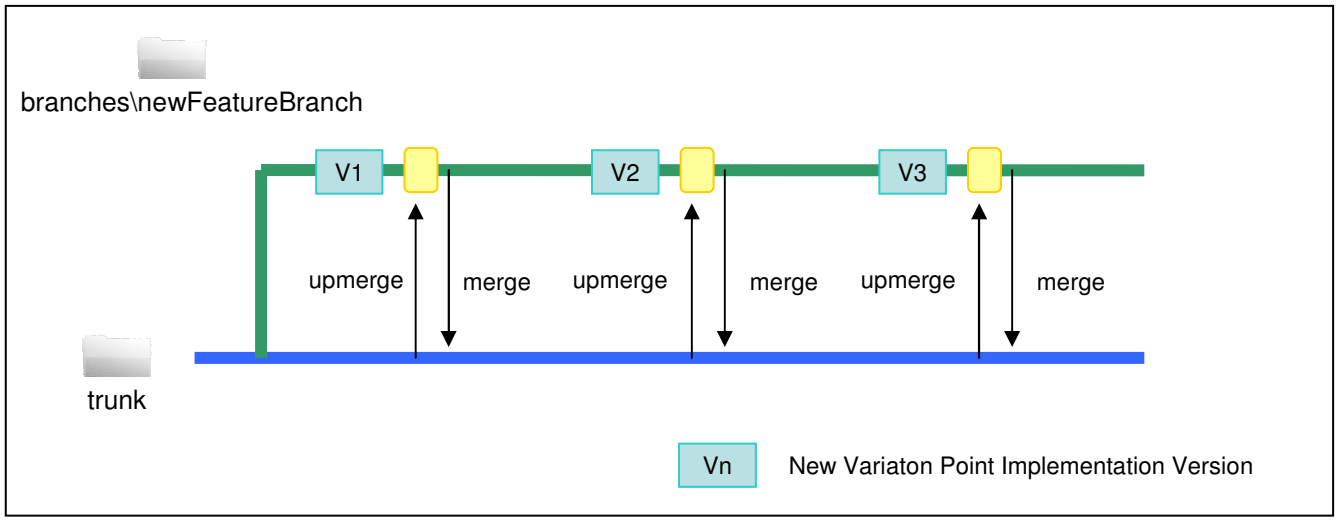

Fig. 4.4 – New variation development on TaRGeT SVN repository

An upmerge should always be done before a merge, to assure that the branch contains an updated version of TaRGeT architecture. After a merge, contributors should send an email to the development team to announce that an updated version of its project is available.

Merge should be done frequently during the project's lifecycle. Continuous merge also decreases integration risk, because developers can identify integration problems, often the most complex to address, during early iterations of software integration.

4.4.2 Modifications on TaRGeT Architecture

During the project lifecycle, contributors may have the necessity to modify the architecture of the TaRGeT application to support the implementation of new features. As TaRGeT is a SPL, these changes cannot be done unilaterally. Contributors must have in mind that their changes may affect other products in (or already out of) the production pipeline.

Modifications on TaRGeT architecture are a result of all parties (contributors and development team) adhering to and agreed-upon process for making any changes. This process aims to assure that the product line will be consistent during all phases of development and that it will support all variations of different TaRGeT products.

In most cases, modifications on the architecture will result on the creation of new variation points, which are intended to capture variability on SPL artifacts. Requests of changes on TaRGeT architecture have to follow the sequence of activities defined in the diagram below:

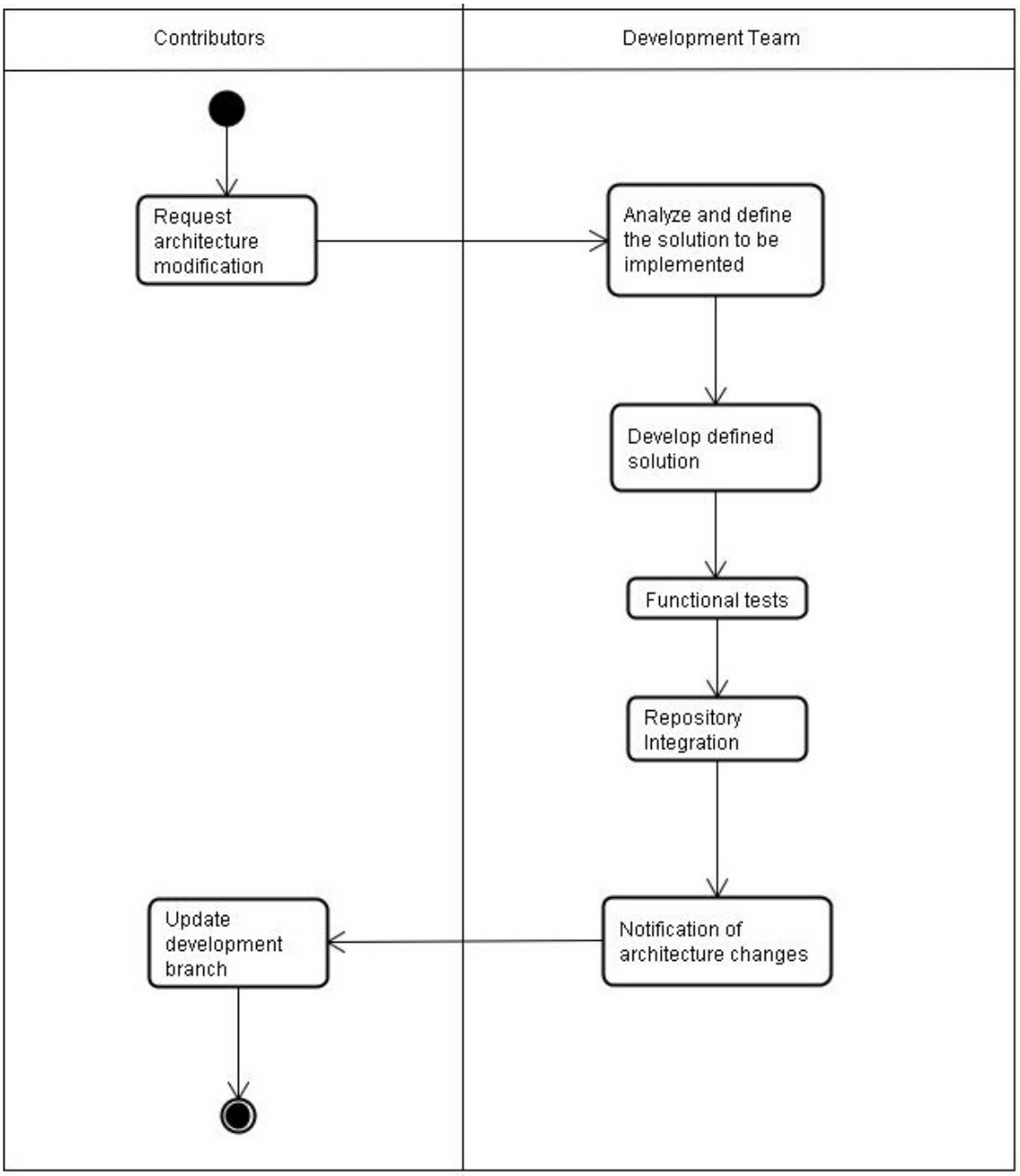

Fig. 4.5 – Sequence of activities for architecture change requests

The picture below illustrates the activities above related with the SVN repository.

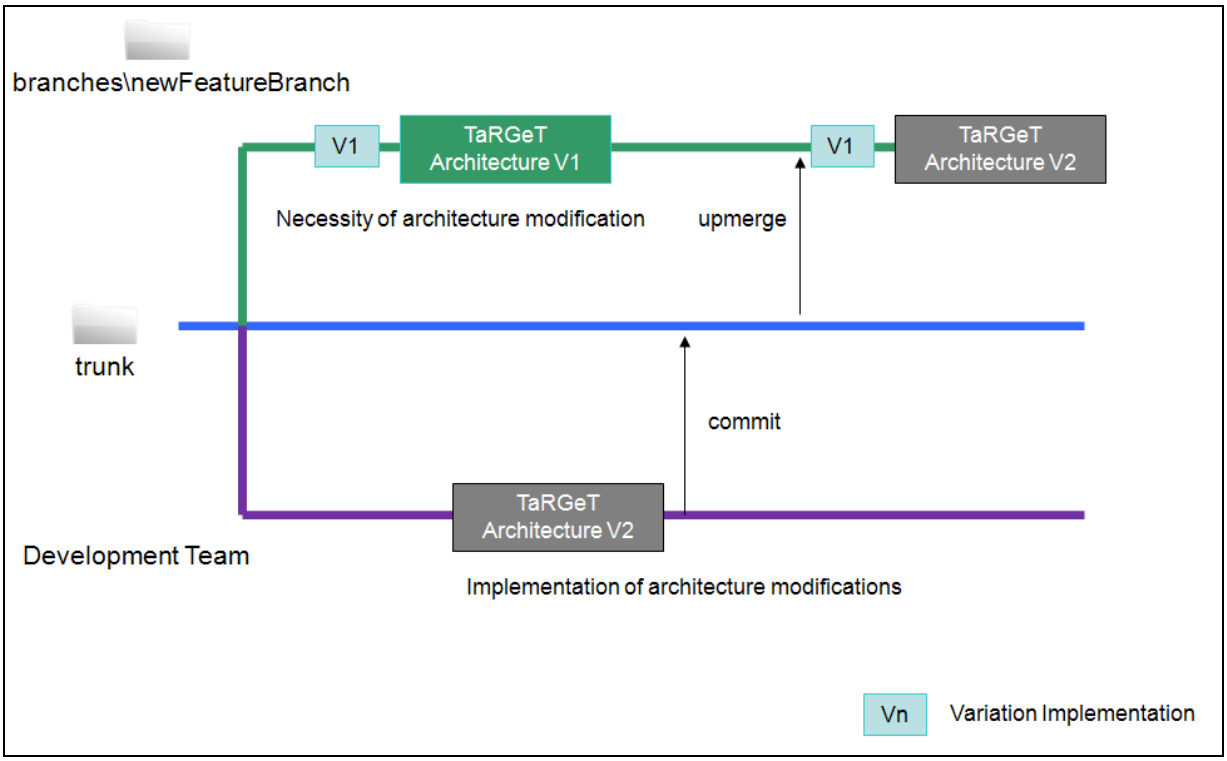

Fig 4.6 – Architecture changes on SVN repository

4.4.3 Bug Correction

TaRGeT has a bug repository to help to keep tracking of the bugs found in the tool.

Reports of new bugs on TaRGeT plug-ins have to be submitted to the bug repository and wait the validation of the development team. The reference [12] contains useful tips on writing quality bug reports.

If the contributor has developed a solution for an opened bug, he has to post the correction on the bug repository in order to have them updated in the repository trunk. The corrections must contain the following fields:

- Branch name: the name of the branch where de correction was developed
- Proposed solution: a description of how the solution was implemented
- Affected plug-ins and classes: a list of the plug-ins and its respective classes that were created or modified and also

the region in the source file that was modified should be indicated using the following notation:

```
//INSPECT – siteName_DeveloperLogin – Small Description
      Code modifications …
//End of inserted/modified code
```

The diagram below summarizes the main activities involved in the submission of bug corrections by contributors:

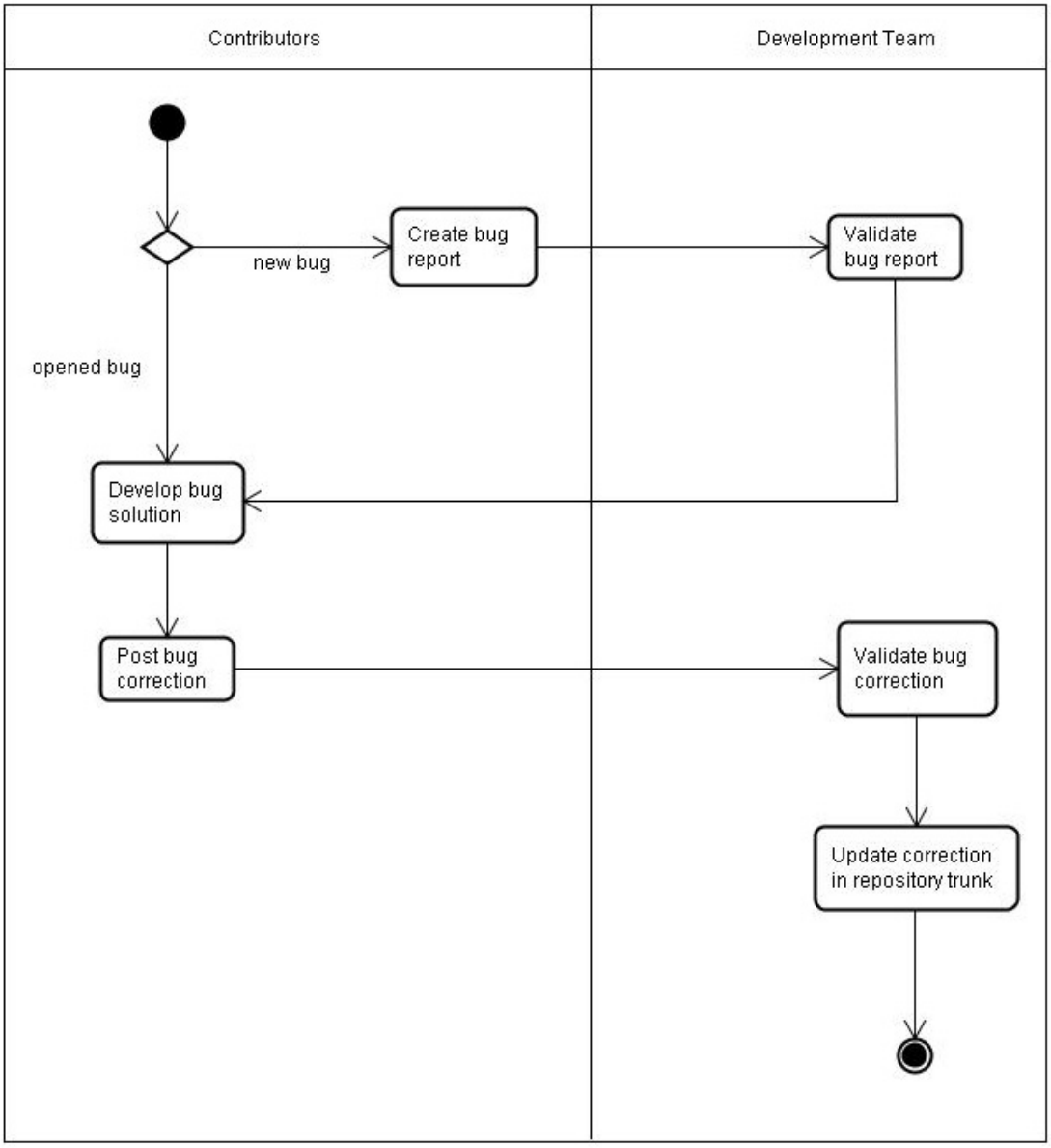

Fig 4.7 – Bug correction submission activities

5. Other Development Artifacts

This section defines the process related to the following artifacts:

- Use Case document;
- Requirements document;
- Architecture document;

According to the different roles involved with the TaRGeT development process.

5.1 CIn Development Team Responsabilities

The development team will be responsible for the maintenance of all documents related to the plug-ins that integrate TaRGeT architecture.

The team is also responsible to update the documentation of TaRGeT proprietary plug-ins.

5.2 Contributors Responsabilities

Contributors will be responsible for the maintenance of the documents related to their specific projects. These documents should be available in the contributor's SVN branch.

6. Releases

This section describes the process involved in releasing formal and informal version of TaRGeT products.

6.1 Informal Releases

Releases of partial versions of new features or bug corrections are considered as informal releases. All TaRGeT products informal releases shall follow the process below:

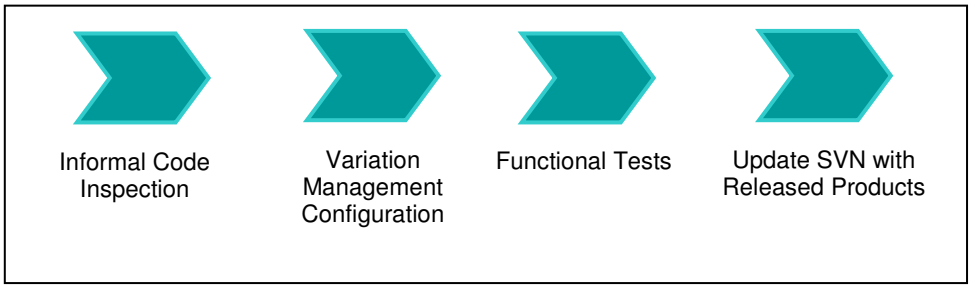

Figure 6.1 – TaRGeT Informal Release Process

- Informal Code Inspection
  - Remove all System.out occurrences
  - Provide Javadoc for methods and attributes
  - Remove any testing and debugging code.
  - Format Code
  - Internationalization support
    - Extraction of GUI text to properties files
- Variation management configuration (Pure Variants)
- Functional Tests

- Update SVN with released products

6.2 Formal Releases

The development team releases one official version of TaRGeT each semester of the year. In this occasion, there will be at least one version of the TaRGeT standard product and other customized products according to different user needs. All TaRGeT SPL formal releases shall follow the process below:

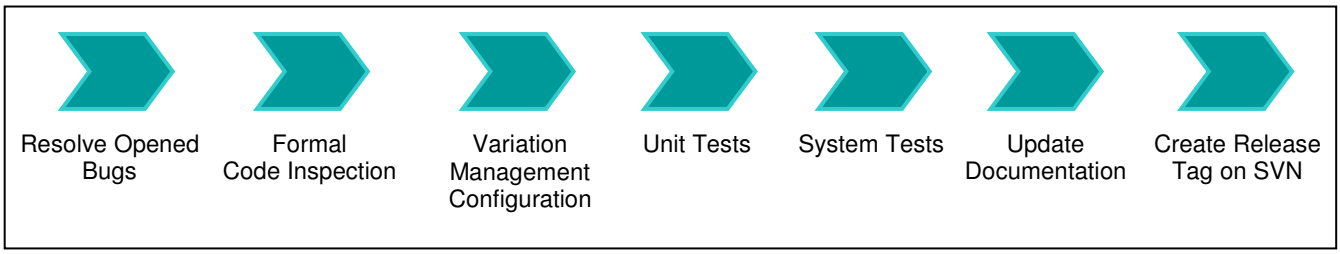

Figure 6.2 – TaRGeT Formal Release Process

- Resolve opened bugs
- Formal Code Inspection
    - Remove all System.out occurrences
    - Provide Javadoc for methods and atributtes
    - Remove any testing and debugging code.
    - Remove INSPECT tags on changes in the code and update classes revision history
    - Format Code
- Variation management configuration (Pure Variants)
    - Update TaRGeT SPL feature model
- Implementation and execution of Unit Tests
- Generate System Tests using the use case document. **All** tests should pass

- Update TaRGeT SPL Documentation
    - Requirements
    - Use Cases
    - Architecture
    - Help Documentation in each plugin
    - Annotate the target.product file with the final version number, date, and time stamps.
- Create the release tag on SVN.
    - Update SVN with released products

7. TaRGeT License

TaRGeT is an open source software and has its code and documentation covered by MIT license [9].

Before talking about the license, let’s define some terms that will be used:

– Distributor: the developer team of the software that give the rights to someone else.

– Receptor: someone that receives the rights to copy, modify and distribute the software.

This license was chosen because it assures copyright for receptors to use TaRGeT and allows them to choose what kind of software (proprietary or free) they will develop based on the tool.

MIT license defines terms and conditions for copying, distribution and modification performed on TaRGeT. Its text does not have copyright, so it can be modified to meet the needs of any project.

According to this license, receptors have the same rights of distributors, so if some functionality is added to the library, its code must be available to receptors to be copied, modified or distributed.

Following this reasoning, all derivative work made by receptors, such as modifications in the architecture or new features developed based on TaRGeT, must be available to the TaRGeT development team. The derivative work can be copied and modified only for research purposes, and it will be distributed only under the developers’ permission.

All license text is available at SVN [11].For any doubt contact development team.