

pvsmlm: A Library Manager for the Prototype Verification System

Miguel Romero¹, Camilo Rocha¹, and César Muñoz²

Escuela Colombiana de Ingeniería, Bogotá, Colombia
NASA Langley Research Center, Hampton VA, USA

Abstract.

1 Introduction

2 PVSLM

This section describe the design and architecture of PVS Library Manager.

Before to describe and analyze the tool it is necessary to define the structure used. The smallest part are the theories, which are organized in packages. Those packages are main element for the tool. Finally there are sets of packages, called libraries. Furthermore, it is necessary to specify the connections or relations within a library, these are the dependencies between its packages. So every package must have a dependencies list defined as the union of the dependencies of its theories.

PVSLM has been developed to manage the installation and update of packages from a PVS library. It has been divided into two sections, the first one manage the source and repository for each library, and the second one manage the packages.

The source section is divided into two parts: the configuration file and the repository. The configuration file contains the information about the library as the name, description and the URL for the git repository where it can be download. It is important to recall that the library must be available on a git server as GitHub or BitBucket. The repository is a clone of a git, that can be updated or deleted. This section is completely dedicated to the configuration of the libraries.

The other section manage the packages, their uninstall, update, list and installation, based on the already configured libraries. PVSLM is based on the dependencies graph of the library, therefore all the modifications on a package will affect the adjacent ones.

As the tool has been developed for the PVS NASA library (nasalib), this one is configured by default. All the commands available on PVSLM are listed along with their description on table 1.

PVSLM works locally, but it needs Internet connection for its installation and the communication with the git repositories. Basically it is downloaded and

Section	Command	Parameters	Description
src	-a	name description URL	Create a new .list file with the library information.
	-d	name	Delete the library .list file.
	-c	name	Create the library repository based on the source with the same name. The creation is a clone of the git repository.
	-u	name	Update the library using a pull command.
	-r	name	Remove the repository.
pkg	-i	library@package	Install and update the package and all its dependencies.
	-u	library@package	Update an installed package and all its dependencies.
	-d	library@package	Delete an installed package with all ones that depends on it.
	-l		List all the installed libraries.
	-l	library	List all the available packages of the library.
	-l	library@package	List all the dependencies of the package.

Table 1: PVS Library Manager command list

then configured, once it is installed there are several commands that needs the to connect to the git repository either to clone or pull it. For example, when a package is going to be installed, it is updated along with all its dependencies. The figure 1 shows how the tool works.

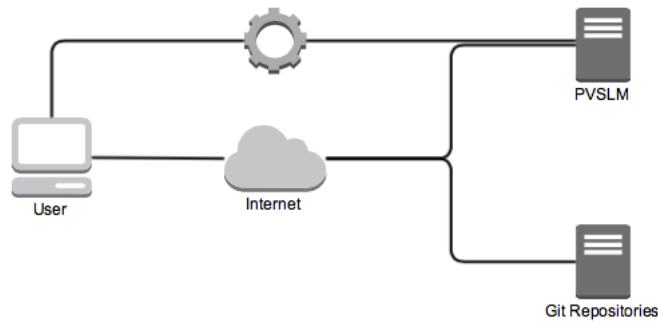


Fig. 1: PVS Library Manager architecture

3 Library Configuration

Requirements

BNF: Dependencies syntax

2 pages

4 Case study: Nasalib configuration and installation

2-3 pages

5 Concluding Remarks