PRT455 – Project Plan

Samuel Walladge (s265679)

August 2017

Contents

1 Introduction

This document is a project plan outlining a proposed software product, a configuration files management program. It will cover background information, a description of the proposed software, its requirements, and associated risks.

2 Background

Many software developers and system administrators have a set of configuration files (dotfiles) for their workstations and servers, fine tuned over many years for the perfect experience. This includes custom shortcuts, text editor configuration, shell prompts, and colour schemes. Generally, these are stored together in a Git repository to back them up, share between computers, and track changes. (Netherland and Jahnke 2017) Below are some examples of such repositories:

- My personal configuration files: `https://github.com/swalladge/dotfiles`
- Those of Greg Hurrell, a developer working for Facebook: `https://github.com/wincent/wincent`
- A popular set of dotfiles by Thoughtbot, with over 4000 stars on GitHub: `https://github.com/thoughtbot/dotfiles`

Also many companies have a shared set of configuration files for developers to use. This helps when doing things like pair programming, as everything is familiar with the setup.

These cannot be used with a system directly. Programs require these configuration files to have certain names and be in certain locations. For example, Vim requires its main configuration file to be at `~/.vimrc` and other configuration in a particular layout under the `~/.vim/` directory. This can't happen when all the files are stored in a single repository.

To keep system configuration files in sync and mapped to files in the repository, a separate program must be used.

2.1 Existing solutions

There are many existing solutions for managing configuration files stored in a repository. (Netherland and Jahnke 2017) These range from huge automation products such as Ansible (Red Hat 2017), to small purpose built programs such as Dotbot (Anish Athalye 2017).

Some are better for different situations, some have more features, others are better tested. They all fall short in some aspect or another.

Personally I have tried custom bash scripts, Dotbot, Ansible, SaltStack (SaltStack Inc. 2017), and finally rcm (Thoughtbot 2017). These cover a broad range of what is available.

Custom bash scripts fall into the category of software custom built for a particular situation. It has the advantage of being tailor built exactly to your use case, but the disadvantage of being slow and difficult when you need more features (you have to program everything yourself).

Ansible and SaltStack are examples of large programs being bent to perform dotfiles management. They are extremely flexible and powerful, but require a lot of configuration themselves to work properly, and slow to set up, and in general aren't optimized for working with dotfiles.

Dotbot and rcm are in the category of community developed purpose built software for managing dotfiles. They are probably the best options available at this time for most use cases, being purpose built for working with dotfiles and flexible since they are designed to work with anyone's repository of files. The down side of these are that they are generally very simple and doing anything more complex is impossible, they often aren't well tested, and they are written in languages like Bash, Python, or Perl that are decent programming languages but prone to having bugs relating to obscure features that are difficult to find.

3 Description of proposed software

This proposed software will aim to provide a configuration files management solution, while solving the problems related to current available software.

This can include:

- writing the software in a safe, compiled language, such as Rust or Go
- using a test driven development style with extensive regression tests to ensure stability
- researching current solutions and finding problem spots
- finding a good balance between flexibility, power, and user friendliness
- include good documentation so users know exactly how it will behave

It can be modelled off existing solutions, being picky about what features to include, what features to take and modify before including, what ones to reject.

The main functions will be to work with a git repository of files and link config files between the computer's expected locations for them and the repository. Other features will be described in the Requirements section.

4 Requirements

This section will includes some high level requirements. These are initial, high level, non-exhaustive lists of planned requirements based on initial research and planning.

4.1 Functional requirements

- It shall be able to work with a single git repository of configuration files. This is the standard method of tracking and syncing dotfiles.
- It shall handle linking configuration files and directories between correct locations and files in the repository.
- It shall provide user friendly methods for adding and removing configuration files from the repository and/or computer.
- It shall be flexible to allow for different styles of storing dotfiles in a repository.
- It shall allow for test runs for actions - to show what files would be affected before actually performing actions.

- It shall provide functions for backing up files that will be overwritten by linking new files.
- It shall provide functions for performing custom actions or scripts when adding, updating, or removing dotfiles.
- It shall provide a simple method for configuring the software itself.
- It shall provide functions for selecting a subset of configuration files for purposes of per-machine config or custom deployments.

4.2 Non-functional requirements

- It shall be consistent. It should never crash and leave the system in a partially modified or broken state.
- It shall be reliable.
- It shall be safe. There should be no possibility of the program deleting any configuration files without backups.
- It shall be light on resources. It should run fine on low-spec computers, small computers, and embedded computers alike.
- It shall require a minimum of dependencies required for installation.

5 Risks

The main risk associated with this is that it fails to produce a useful solution altogether. Many other config management programs have been built and have problems or aren't that great. This raises a question: why should this succeed when others have failed? To mitigate this risk, good software develop practices must be followed. With a clear goal, good plans, and test driven development, there is no reason why it shouldn't succeed.

To avoid developing software that has a bad api or is not user friendly, user feedback and prototyping must be included in the development process.

Since this will be a program that works on files in a user's home directory, there is the potential for bugs or unforeseen operations to result in data loss. The legal implications of this must be accounted for, and appropriate disclaimers or licenses used.

References

Anish Athalye. 2017, *Dotbot*, version 1.11.1, viewed August 1, 2017, <https://github.com/anishathalye/dotbot>.

Netherland, W and A Jahnke. 2017, *GitHub does dotfiles*, viewed August 1, 2017, <http://dotfiles.github.io/>.

Red Hat. 2017, *Ansible*, viewed August 1, 2017, <https://www.ansible.com>.

SaltStack Inc. 2017, *SaltStack Documentation*, SaltStack, viewed August 1, 2017, <https://docs.saltstack.com/en/latest/>.

Thoughtbot. 2017, *rcm*, version 1.3.1, viewed August 1, 2017, <https://github.com/thoughtbot/rcm>.