BACHELOR SOFTWARE ENGINEERING PROJECT

Software Transfer Document

Team Waterfowl

July 1, 2021

DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

Eindhoven University of Technology

2IPE0 Software Engineering Project
QUARTILE IV - 2020-2021

Software Transfer Document

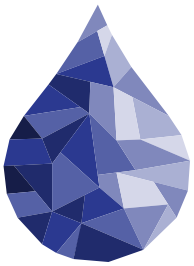
Team Waterfowl

Authors:
Adrien Castella *(1280880)*
Adrian Cucoş *(1327860)*
Cosmin Manea *(1298542)*
Noah van der Meer *(1116703)*
Lulof Pirée *(1363638)*
Mihail Ţifrea *(1317415)*
Tristan Trouwen *(1322591)*
Tudor Voicu *(1339532)*
Adrian Vrămuleț *(1284487)*
Yuqing Zeng *(1284835)*

Supervisor:
Gerard Zwaan, univ. lecturer
Customer:
Jim Portegies, asst. prof.

Version 0.2

July 1, 2021



Abstract

This document describes the Software Transfer Document (STD) for Waterproof. It contains the relevant information to build all the software artefacts produced by Team Waterfowl along with steps for continuing the development of the project. Furthermore, problems, requested changes, and modifications done in the transfer period are explained. This document complies with the ESA software standards (cf. [9]).

Contents

Document Status Sheet

Title	Software Transfer Document
Authors	Adrien Castella, Adrian Cucoş, Cosmin Manea, Noah van der Meer, Lulof Pirée, Mihail Ţifrea, Tristan Trouwen, Tudor Voicu, Adrian Vrămuleț, Yuqing Zeng
Version	0.2
Status	Final
Creation Date	May 17, 2021
Last Modified	July 1, 2021

Document History log

June 30^{th}, 2021: First version release (version 0.1).

July 1^{st}, 2021: Final revision (version 0.2).

1 Introduction

1.1 Purpose

The purpose of this Software Transfer Document is to familiarise future developers with the development process of the Waterproof installer, the tactics library, and the abstract syntax tree parser with their build procedure, installation procedure, configurations, testing procedure, and results. It will also include information on present bugs in Section 6. During the transfer of the software, change requests will come up and some modifications might get made. The change requests will be in Section 7 and modifications to the code will be recorded in Section 8.

1.2 Scope

The Waterfowl project of improving Waterproof is being developed by a group of Computer Science students from Eindhoven University of Technology, as part of their Software Engineering Project. The goal of the project is to improve the user experience of the mentioned application, as well as to facilitate updating and performing maintenance on it.

Currently, since Waterproof is built on top of the opam system (cf. [7]), maintenance and installation are complex tasks to perform. The tasks are to improve the user experience and maintainability of Waterproof.

In achieving this goal, an easy installer, whose purpose is to set up the underlying ecosystem, as well as an updater have to be created. Moreover, improvements to the Waterproof application have to be made. The most important ones are about the UI and its dependencies. To this end, syntax highlighting and a custom tactics library have to be implemented.

1.3 List of definitions

Term	Definition
Coq	A formal proof management system that allows for expressing mathematical expressions and assertions (cf. [5]).
Coq-SerAPI	A library for machine-to-machine interaction with the Coq proof assistant.
Cygwin	A program allowing the user to run native Linux applications on Windows.
Electron	A framework for creating cross-platform desktop applications using web technologies (JavaScript, HTML, and CSS).
Executable file	A program that can be run in the OS without explicitly needing to load it with another program.
Git	A version control system, a tool used to manage and track changes of software projects.
Mechanization	Automating a process for more efficiently accomplishing a given task.
GitHub	A provider of Internet hosting for software development, using Git.
OCaml	A general-purpose typed programming language. OCaml is designed to enhance expressiveness and safety (cf. [6]).
Opam	The OCaml Package Manager (cf. [7]). A source-based package manager for distributing OCaml programs and tools.
Proof assistants	"computer programs" specifically designed for mechanizing rigorous mathematical proofs on a computer.
Pull request	A method to propose changes to the Git repository of a code project.
Software dependency	A software component necessary for the operations of a different program.
Software package	A collection of applications or code modules that work together to meet various goals and objectives.
Software repository	A storage location for software packages.
Syntax highlighting	The feature displays text, especially source code, in different colours and fonts according to the category of terms.
User	Person that uses the application.

1.3.1 Abbreviations

API	Application Programming Interface
AST	Abstract Syntax tree
JS	JavaScript
GB	Gigabytes
OS	Operating System
PDF	Portable Format Document
RAM	Random-access memory
SEP	Software Engineering Project
UI	User Interface
URL	Uniform Resource Locator

1.4 List of references

[1] Adrien Castella et al. *Waterfowl Acceptance Test Plan*. Eindhoven University of Technology, June 2021.

[2] Adrien Castella et al. *Waterfowl Software Design Document*. Eindhoven University of Technology, June 2021.

[3] Adrien Castella et al. *Waterfowl Unit Test Plan*. Eindhoven University of Technology, June 2021.

[4] Adrien Castella et al. *Waterfowl User Requirements Document*. Eindhoven University of Technology, June 2021.

[5] Thierry Coquand, Gérard Huet, and Christine Paulin. *The Coq Proof Assistant*. Accessed: 2021-04-27. URL: https://coq.inria.fr/.

[6] *OCaml*. Accessed: 2021-04-27. URL: https://ocaml.org/index.html.

[7] *OCaml Package Manager*. OCamlPro. URL: https://opam.ocaml.org/ (visited on 06/26/2021).

[8] Jim Portegies et al. *Waterproof Tactics Library*. URL: https://github.com/impermeable/coq-waterproof (visited on 06/29/2021).

[9] ESA Board for Software Standardisation and Control (BSSC). *ESA Software Engineering Standards: Issue 2*. 1991.

[10] *Waterfowl Software User Manual*. Eindhoven University of Technology, 2021.

2 Build procedure

The software build process was split along the three main software artefacts, namely the installer, the AST parser and the tactics library.

While no continuous integration environment was used, every major change was validated through unit, and manual tests before being merged.

In terms of build time, on a Windows machine with 8GB of RAM and a 4 Core i3-8300, the installer took roughly 1 hour and 9s to build, the new Waterproof software containing the AST-parser and the updater took 4.5 minutes for a clean build and the tactics library was compiled in under 3 minutes and 2 seconds. As these are integrated into the existing Waterproof code, the build times of these two components cannot be measured independently.

No problems were encountered during the build, using the default configurations mentioned in the following subsections.

2.1 Automatic NSIS installer build procedure

The build procedure is automated using a GitHub workflow. The workflow runs automatically when pushing to the GitHub repository at

https://github.com/impermeable/waterproof-dependencies-installer.

The developer must navigate to the Actions page and open the most recent workflow named 'Windows'. The 32-bit and 64-bit will be available for Download for 5 days after the build under the Artifacts tab.

2.2 Manual NSIS build procedure

It is also possible to manually perform the build. This can be convenient for quicker debugging since every change in the code will require the complete Workflow to be repeated, which takes a long period of time. By only repeating the steps which are incorrect a lot of time can be saved. We assume the developer has a Windows environment with a version of git installed in the command prompt.

2.2.1 Setting up Coq platform environment

Before building, we need to set up the Coq platform environment. This is a Cygwin environment where Opam and Coq are installed. To create this environment we will use a script from the Coq platform project. The setup starts by cloning the Coq platform repository. It is important that automatic line ending conversion is turned off. Otherwise, Cygwin errors will occur regarding Windows line endings in the Cygwin code. Alternatively just download and unzip the zip file of the project.

```
$ git clone --branch 2021.02 https://github.com/coq/platform.git
```

Navigate to the cloned platform's folder in the command prompt:

```
$ cd platform
```

In this folder, there is a `batch` file `coq_platform_make_windows.bat` that needs to be executed. The script will guide you through a process where the installation location should be and which Coq libraries should be installed. Select the minimum number of packages (the basic install) by typing 'b'.

Alternatively, the following command can be used such that no interactivity is required:

```
$ coq_platform_make_windows.bat -destcyg=<path>
 -cygrepo=<cygwin download repository> -build=y -cygquiet=y -cygforce=n
 -extent=f -compcert=n -vst=n -switch=k
```

Here `<path>` is an install location for Cygwin, and `<cygwin download repository>` is the mirror link. In the Netherlands, the preferred Cygwin download mirror is `https://mirror.koddos.net/cygwin/`. More can be found at `https://cygwin.com/mirrors.html`. This choice improves the download speed.

A list of Coq platform setup parameters and their usage can be found in Table 1.

After the execution of the command from above, you will be prompted with a confirmation of the parameters. Confirm the choice of parameters by typing *y* and then *ENTER*.

flag	explanation
-extent=f	Setup opam and build full Coq platform.
-extent=b	Just setup opam and build Coq (basic).
-extent=i	Just setup opam and build Coq + CoqIDE.
-parallel=p	Build several opam packages in parallel.
-parallel=s	Build opam packages sequentially.
-jobs=1..16	Number of make threads per package.
-compcert=f	Build full non-free version of CompCert.
-compcert=o	Build only open source part of CompCert.
-compcert=n	Do not build CompCert and VST.
-cyglocal= y or n	Set whether to install cygwin from cache.
-vst=y	Build Verified Software Toolchain (takes a while).
-vst=n	Do not build Verified Software Toolchain.
-switch=k	In case the opam switch already exists, keep it.
-switch=d	In case the opam switch already exists, delete it.

Table 1: `coq_platform_make_windows.bat` parameters

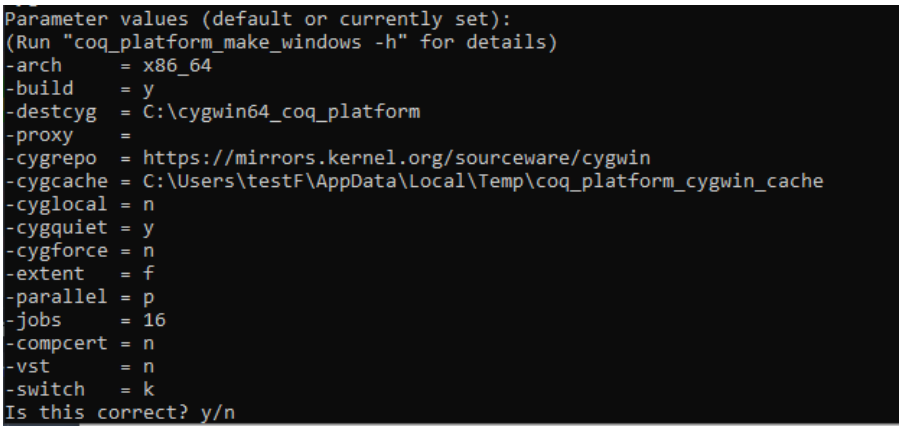


Figure 1: Environment diagram

Shortly after, a Cygwin download window will pop-up and start downloading Cygwin on your computer. This is expected behaviour.

When the platform is installed navigate to the installation folder and right-click `cygwin.bat` and click run as administrator. A Cygwin terminal will pop up. We will call this terminal the Coq platform environment.

2.2.2 Installing Opam packages

In the Coq platform environment, packages can be installed using

```
$ opam install <package>
```

This is also how Coq-SerAPI is installed. Hence, Waterproof can be used using this installation.

2.2.3 Installing GitHub Coq packages

GitHub Coq packages can be easily installed in the Coq platform environment by cloning the repository, entering the repository folder and running the following

```
$ make
$ make install
```

2.2.4 Building installer

The installer executable is made by running the `create_installer_windows.sh` file. Note that this must be done from the `coq-platform` folder. Hence, the following commands will build the installer:

```
$ cd coq-platform
$ ./windows/create_installer_windows.sh
```

This will create an installer with the packages installed with opam.

2.2.5 Patching Coq-SerAPI

Coq-SerAPI is not supported by the Coq platform. Several fixes are needed for it to run. To make sure Coq-SerAPI works, the script `patch_serapi.sh` from the GitHub repository with the following URL must be run:

https://github.com/impermeable/waterproof-dependencies-installer

Make sure the `patch_serapi.sh` is in the home directory, so not `coq-platform`. Then make it executable and run it as follows.

```
$ chmod +x patch_serapi.sh
$ ./patch_serapi.sh
```

2.2.6 Installing additional packages

In addition to the patch for Coq-SerAPI, the waterproof-dependencies-installer repository contains the script `install_packages.sh` which extends the `create_installer_windows.sh` file with code to automatically install the packages in the configuration file `packages.cfg`. For info on this configuration file, see Section 2.2.7. Make sure all the files of the repository are in the `coq-platform` folder and run

```
$ cd coq-platform
$ chmod +x install_packages.sh
$ ./install_packages.sh
```

Notice that this permanently changes the `create_installer_windows.sh` file. Hence, it is wise to create a backup if needed.

2.2.7 Using the configuration file

The repository contains a configuration file called `packages.cfg` that specifies which opam packages need to be installed. It also specifies Github repositories containing Coq libraries to be installed. In addition, each package has an option to be installed by default in the installer or not.

The configuration file contains two types of lines.

Firstly, for a custom Github package, six items need to be specified; each of them between quotes (e.g. ""):

1. Package name: the name of the package
2. Installed by default in installer (SELECTED/UNSELECTED)
3. Relative package location: the path location in windows format relative to the Opam switch
4. Package description: textual description shown in installer
5. Github owner: repository owner
6. Github repository: name of the repository

The opam switch folder is the folder containing all files installed by opam and is located in the folder `/.opam`.

The syntax is then:

```
GITHUB "[package name]" "[selected]" "[relative package location]"
"[package description]" "[Github owner]" "[Github repo]"
```

The second type of line is for Opam packages. The following argument should be specified for this:

1. Package name: the name of the Opam package
2. Installed by default in installer (SELECTED/UNSELECTED)

The syntax is:

```
OPAM "[package name]" "[selected]"
```

2.3 Releasing Updates

In order to release an update to GitHub that will be recognised by the updater integrated into Waterproof, the deployment script provided in the Waterproof codebase should be used. First, request an API token for the GitHub account on which the software should be released.

If you are using Microsoft Windows, open a command prompt and run the following command:

```
$ set GH_TOKEN=YOUR_TOKEN
```

where you replace *YOUR_TOKEN* with the API token obtained earlier from GitHub. If you are using Linux or MacOS, open a bash terminal and use the command

```
$ export GH_TOKEN=YOUR_TOKEN
```

where you replace *YOUR_TOKEN* with the API token obtained earlier from GitHub. Next, navigate to the main Waterproof directory in your console and run the command

```
$ npm run deploy
```

This will build the software for the operating system you are using and creates a release draft on GitHub on which the locally built executables are uploaded. This procedure should be performed for all platforms for which the update is intended.

In order to release the update to the public, you can now simply publish the release on GitHub. The update will be recognized by all newly started Waterproof clients running the target platforms, and by running Waterproof clients once they have been restarted. The Waterproof updater is based on the electron-updater package. See the official documentation of the electron-updater package for further details.

2.4 AST-parser and Syntax highlighter Build procedure

As the AST-parser and the syntax highlighter are closely integrated with the existing Waterproof software project, their build procedure is the same as that of the original project.

Assuming a system on which Waterproof was not previously installed, but that already has the necessary build dependencies, namely git and NodeJS, the build procedure is as follows:

1. Clone the Waterproof software from Gitlab using git:

   ```
   git clone https://gitlab.tue.nl/sep-group-2-2020-2021-q4/waterproof.git
   ```

2. Following a successful clone, navigate to the newly created waterproof folder.
3. From the context of this folder, execute the command

   ```
   npm install
   ```

   to download all the project-specific dependencies.
4. Once this project is completed, we can build
   (a) a development version using

   ```
   npm run electron:serve
   ```

   (b) or a full, install-able build using

   ```
   npm run full-build
   ```

   in which case the result will be an platform-specific executable.

2.5 Tactics library build procedure

The new tactics library is a collection of Coq files. In order to separately build the library, one only needs to use a version of the Linux operating system that supports the Coq proof assistant (such as Ubuntu 20.04), and to perform the following steps:

1. Clone the GitHub repository

   https://github.com/impermeable/coq-waterproof

   on the local machine, in a particular folder. We will refer to this folder as `branch`.
2. Open the Linux terminal and go to the folder `branch`.
3. Run the command `make`.

When the above Linux command finishes, the tactics library is fully compiled and hence built.

3 Installation procedure

The Waterfowl project is cross-platform. While the installer is Windows-specific, all the other components developed during this project, namely the tactics-library, the AST-parser, and the updater are compatible with Windows, Mac OSX, and Linux.

3.1 Using the Waterfowl installer

On the Windows 10 OS, all the components of the Waterfowl project can be installed by using the installation scripts developed by the Waterfowl team.

After running the executable, Waterproof and the tactics library, along with all their dependencies will be installed on the user's computer.

The updater is inherently part of the Waterproof application itself, so it does not need a separate installation procedure.

3.2 Abstract Syntax Tree parser

There is no additional installation required for the AST parser. The cloning of the GitHub repository and the building with `npm` together with SerAPI already provide everything that is necessary.

3.3 Tactics library

As previously mentioned in Section 3.1, on the Windows 10 OS, the tactics library is installed through the Waterfowl installer.

However, it is possible to install the tactics library on any Linux OS that supports the Coq proof assistant. This is due to the fact that the tactics library has been released as an official opam package (cf. [8]).

In the remainder of this section, an installation procedure for the tactics library on the Linux OS will be given. It is assumed that the opam software (cf.[7]) and the Coq proof assistant are already installed on this machine.

The steps for installing the tactics library are the following:

1. Open the Linux terminal.
2. Run the command

   ```
   opam repo add coq-released https://coq.inria.fr/opam/released
   ```

   and wait for it to finish.
3. Run the command

   ```
   opam install -v coq-waterproof
   ```

   and wait for it to finish.

The last command from above installs and compiles the tactics library completely.

4 Configuration item list

This section provides an overview of all artefacts that will be delivered to the customer.

The items under specified in Sections 4.1 and 4.2 are documents (in PDF format) of the Waterfowl project (the Software Design Document, the User Requirements Document, the Software User manual, the Acceptance Test Plan, the Unit Test Plan, and this document, the Software Transfer Document). These were transferred to the customer via email, and they can also be found on the GitHub pull request, at

`https://github.com/impermeable/waterproof/pull/23`.

The items under Section 4.3 were transferred as follows:

- The installer software can be found on the GitHub repository

  `https://github.com/impermeable/waterproof-dependencies-installer`

- A pull request has been made for the Updater software and the Abstract Syntax Tree parser software, which can be found at

  `https://github.com/impermeable/waterproof/pull/23`

- The new tactics library can be found on the GitHub repository

  `https://github.com/impermeable/coq-waterproof`

4.1 Documentation

- User Requirements Document (cf. [4]).
- Software Design Document (cf. [2]).
- Software User Manual (cf. [10]).
- Software Transfer Document (this document).

4.2 Test plans

- Acceptance Test Plan (cf. [1]).
- Unit Test Plan (cf. [3]).

4.3 Software

- Installer software: the software designed for installing the back-end dependencies of Waterproof.
- Updater and Abstract Syntax Tree parser software.
- New tactics library software.

5 Acceptance test report

5.1 First acceptance test

The first acceptance test took place on the 17th of June 2021. The customer, the team members and the project supervisor were present.

Out of the acceptance tests present in the Acceptance Test Document (cf. [1]), only 12 acceptance tests were performed (ATP's #3-7, ATP's #13-16, ATP #18, ATP #19 and ATP #23). Out of these, ATP's #3, #5, #7, #13-16, #18, #19 and #23 were deemed a success, but ATP's 4 and 6 were failed. This meant that certain parts of the software artefacts that we developed were working correctly, and certain parts were not.

However, the customer did not fully agree with ATP's #13-16, and requested more visual aids for these acceptance tests.

5.2 Second acceptance test

The second acceptance test took place on the 24th of June 2021. The customer, the team members, and the project supervisor were present.

Certain acceptance tests were redesigned and re-tested in the final version of the Acceptance test plan (cf. [1]). These were ATP's 13-16 (as the customer required).

Out of the acceptance tests present the Acceptance Test Document (cf. [1]), ATP's #18, # 19 and # 23 were not re-tested, as they were deemed a success in the first acceptance test (cf. Section 5.1). The rest of the ATP's were attempted during this phase.

All of these remaining ATP's except ATP #2 were passed by the customer on that respective day, and no additional changes were required.

For ATP # 2 (which was about all software artefacts produced throughout the Waterfowl project are transferred back to the customer via GitHub), the customer agreed to pass this acceptance test and sign the Acceptance Test Plan document accordingly, once all the Updater and the Abstract Syntax Tree software artefacts were in a ready-to-merge state and a pull request was made to the customer's original GitHub repository, once the new tactics library was pushed onto the `coq-waterproof` GitHub repository, and once the installer software was pushed onto the `waterproof-dependencies-installer` GitHub repository. As the Waterfowl team realised this on the 29^{th} of June 2021, ATP # 2 was also passed by the customer, and so all acceptance tests presented in [1] were passed.

6 Software Problem Reports

During the software transfer phase and after the last acceptance test, no software problems were raised, and, as such, there are no software problem reports.

7 Software Change Requests

During and after both acceptance tests, the customer was content with the software artefacts produced by the Waterfowl project, and, as such, there were no software change requests.

8 Software Modification Reports

During the software transfer phase, a few modifications were made to the new tactics library software artefacts. These were:

- In order to avoid code duplication, a small part of the functionality of a tactic was changed so as to use an already existing function. This change can be found at

  `https://github.com/impermeable/coq-waterproof/commit/7f28aeb9955508c2ff8c056a1342ae67a44aaee7.`

- The small description of another tactic was also changed so as to reflect its actual functionality. This change can be found at

  `https://github.com/impermeable/coq-waterproof/commit/f5c2646ace9d2d244768932b8256fbb6ee60dae6.`

The customer agreed with both of these changes.