# Building WorkflowMaker

For WorkflowMaker version **1.1.2** and later
Windows 10 & 11, Ubuntu-based Linux distributions
64-bit only

# Table of contents

# List of figures

# 1 The goal

This document explains the organization of the WorkflowMaker software, describing its distribution across different folders and sub-folders, and indicating what can be found in each one. Additionally, it provides instructions for building WorkflowMaker from the source code, for both Windows and Linux. It specifies the tools to be used; it also indicates the procedure for installing said tools. Finally, some important comments are made regarding the Application Programming Interface (API) documentation.

⚠️ Due to the limited resources available, the build instructions for Linux systems provided in the next sections are guaranteed to work only on Ubuntu-based Linux operating systems. Furthermore, and since version 6 of the Qt framework (see section 2) is required, only relatively new distributions such as Ubuntu 24.10 or Linux Mint 22 are supported; previous ones include version 5 of said framework only.

# 2 The tools

To build WorkFlowMaker from source the following tools are required:

- Windows
  - MSVC 2022 Community Edition (https://visualstudio.microsoft.com/downloads/).
  - Qt version 6 (https://www.qt.io/download-open-source). Download the prebuilt binaries for the MSVC 2022 compiler (msvc19_64).
  - The NSIS (Nullsoft Scriptable Install System (https://nsis.sourceforge.io/Download).
  - The HM NIS EDIT editor (https://hmne.sourceforge.net/). This editor is used to build the Windows installer together with NSIS.
  - doxygen (https://www.doxygen.nl/download.html) and graphviz (https://graphviz.org/) to build the documentation.
- Linux
  - gcc 13.
  - The dpkg and dpkg-dev packages (for building .deb packages and finding package dependencies).
  - Qt version 6.
  - doxygen and graphviz, to build the documentation.

For Linux, all tools should be available using the built-in package manager.

⚠️ On Linux **do not** download version 6 of the Qt libraries from Qt's website; use the ones available in the system repositories instead. Deploying a Linux application relying on the libraries included in the official Qt's distribution is much more complicated than using the system libraries; furthermore, the size of the resulting deb package is notably bigger. See section 4.1.2 for details about installing the necessary tools on Linux.

# 3  The organization of the source code

The first-level sub-folders of WorkflowMaker's main folder is organized as depicted in Figure 1. Their contents is described below;

- `data` – XML schema files defining the syntax of the toolkit, workflow and launcher files.
- `docs` – The documentation, including this guide.
- `installer` – Scripts to build installers, either for Windows or Linux platforms.
- `libraries` – The source code of the libraries developed and used by the tools in WorkflowMaker.
- `QT_redistributables` – The Qt redistributable shared libraries required by the Windows version of WorkflowMaker. Not required for Linux builds.
- `samples` – Data and source code samples for an example toolkit (simple image processing).
- `tools` – The source code for the tools (ToolkitEditor, WorkflowEditor, WorkflowLauncher).
- `VC_redistributables` – Contains a single file, the installer of the Microsoft Visual C++ redistributable files required by the Windows version of WorkflowMaker. Not required for Windows builds.
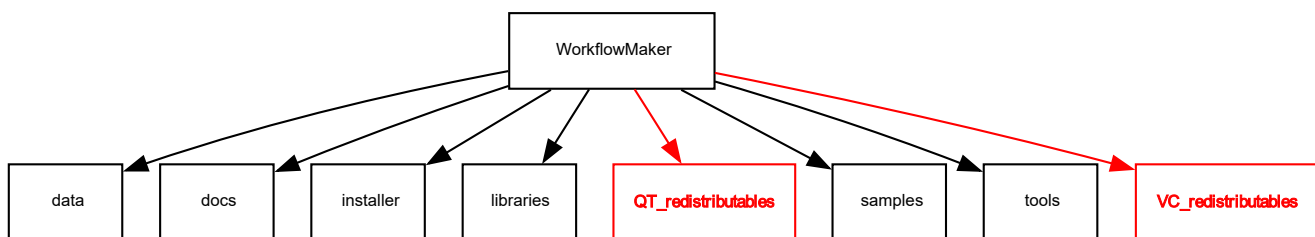


*Figure 1: Folder organization. Folders in red are required only for Windows builds.*

Next subsections describe in more detail the structure and contents of each sub-folder in the list above. Note that the `data` sub-folder is not further explained, since these contain no sub-folders on their own.

## 3.1  The docs sub-folder

The structure of the installer sub-folder is shown in Figure 2.

- `base_icons` – The icons used by the applications and documents included in the WorkflowMaker suite. It is subdivided into two sub-folders: `freepik` and `workflowmaker`. The first one contains base images downloaded from [freepik.com](freepik.com) (see section 5.3 for more details) while the second one stores the icons built using said base images.
- `licenses` – In this folder, a copy of the license files corresponding to the libraries used by WorkflowMaker are stored.
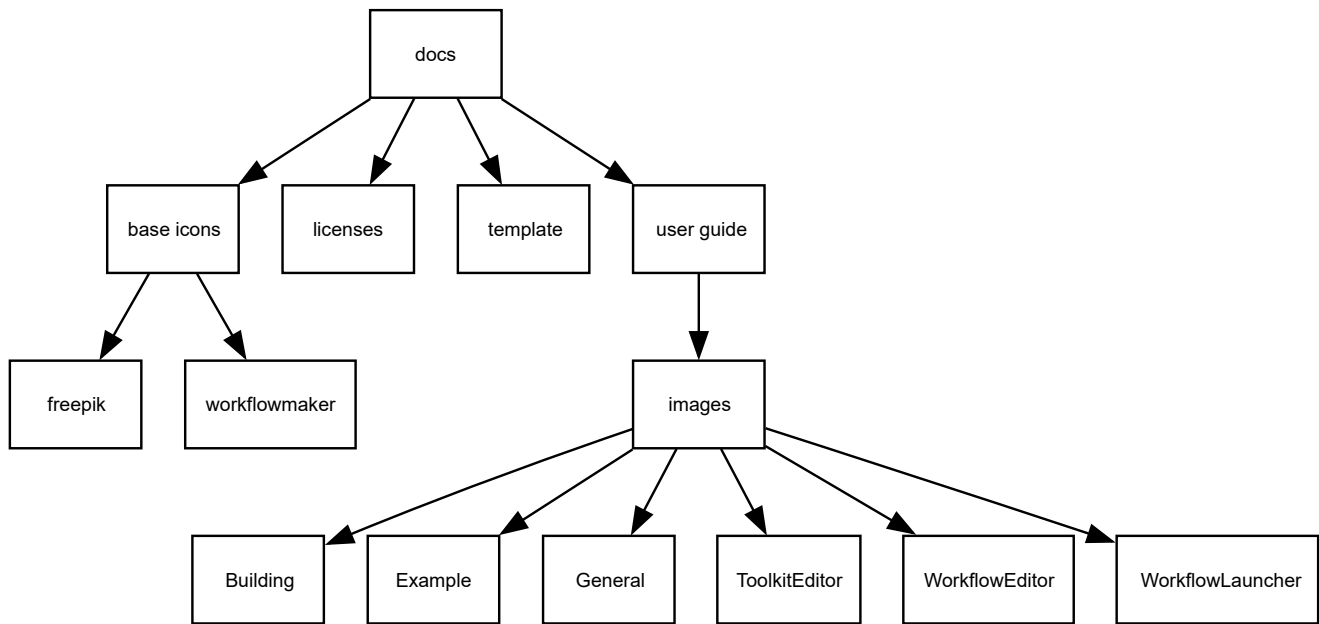
*Figure 2: The structure of the docs sub-folder.*

- `template` – The LibreOffice's Writer template used to write all the documents included in the distribution.
- `user guide` – Here, all the user guides included with WorkflowMaker may be found, either in LibreOffice Writer format or as PDF files. Additionally, there is a sub-folder named `images`. There, the images used by said user guides are stored. It is subdivided into several sub-folders keeping apart the images corresponding to different guides or important sections of these. These sub-folders are:
  - `Building` – Images for the guide named 'Building WorkflowMaker.odt / .pdf'.
  - `Example` – The main user guide, 'WorkflowMaker user guide.odt / .pdf' includes a section with examples. The images used by such section may be found in this folder.
  - `General` – Those images used by two or more guides, such as cover images.
  - `ToolkitEditor`, `WorkflowEditor` and `WorkflowLauncher`. Once more these are images used by the main user guide, namely 'WorkflowMaker user guide.odt / .pdf'. Each sub-folder contains, respectively, images for the subsection dealing with each of the tools whose names stand for.

## 3.2  The installer sub-folder

The structure of the installer sub-folder is shown in Figure 3.

Two different sub-folders exist here.

The first one, `installer_sw_windows` includes all the necessary software and data to build a Windows installer using the Nullsoft Scriptable Install System (NSIS). This includes the NSIS script – the code defining how to build the installer – as well as several images and icons required to build it.

The second sub-folder, `installer_sw_linux`, is targeted at building a package for Ubuntu-based Linux platforms. There, there are three files, namely `control`, `postinst` and `prerm` that will be be copied to the Linux .deb package folder structure, more precisely to the `DEBIAN` sub-folder. These define the characteristics of the package (`control`), the post-build actions to start when the package is copied to the system (`postinst`) and those to execute when uninstalling it (`prerm`). See also section 4.1.
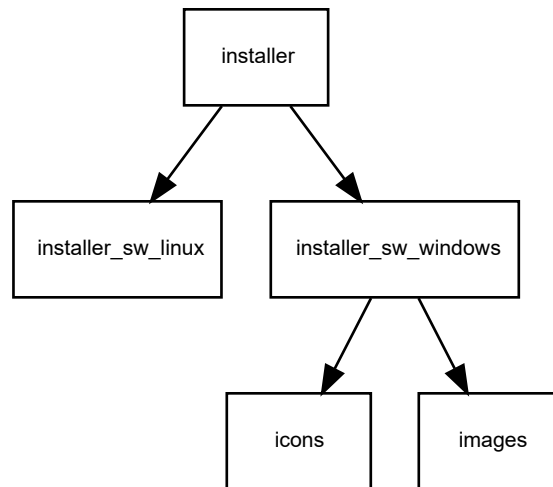


*Figure 3: The structure of the installer sub-folder.*

## 3.3  The libraries sub-folder

Figure 4 depicts the structure of the `libraries` sub-folder.
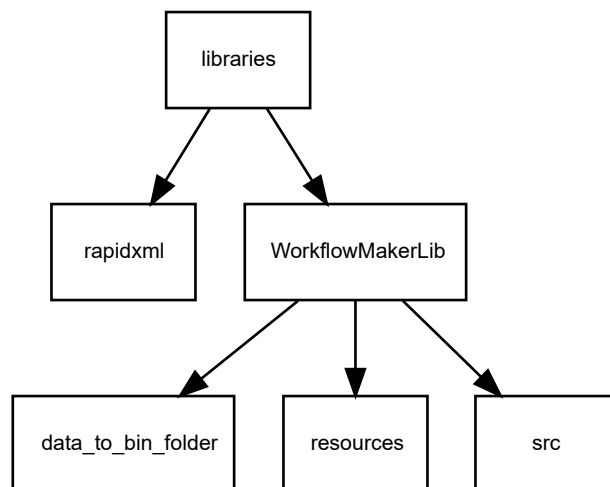


*Figure 4: The structure of the libraries sub-folder.*

This sub-folder must contain all the libraries used by the tools in the WorkflowMaker package, either self-developed or third-party. Currently, there are two libraries, namely `rapidxml` (open source, https://rapidxml.sourceforge.net/) and `WorkflowMakerLib`, which has been developed specifically for this project.

No description is provided for the `rapidxml` library. Being a header-only library, it is sufficient to include those headers in the software that uses said library.

The `WorkflowMakerLib` folder is organized as follows;

- `data_to_bin_folder` – A single file is included here, namely `workflowmaker_version.txt`. It contains the version string for all the applications in WorkflowMaker. Changing this file and rebuilding the library will make all these tools to show a new version string.
- `resources` – Images and HTML files to help build the developer's documentation.
- `src` – The source code for the library.

## 3.4 The QT_redistributables sub-folder

⚠ This sub-folder is only required for Windows builds. When working on Linux Qt must be present as a system's package instead. See section 4.1.2 for details on how to install Qt on Linux.

The `QT_redistributables` sub-folder is structured as shown in Figure 5.
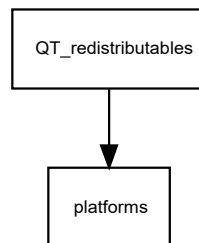


*Figure 5: The structure of the QT_redistributables folder.*

This folder contains all the Qt dynamic link libraries (DLLs) and platform code required by the applications. These are:

- Root:
  - `Qt6Core.dll`
  - `Qt6Gui.dll`
  - `Qt6Widgets.dll`

- Sub-folder `platform`
  - `qwindows.dll`

*All these dynamic libraries are copied from the Qt installation folder* (see section 4.2.1 which explains how to install Qt on a Windows platform). This is typically `c:\Qt`. Assuming that this is the root directory, then all the DLLs that must be copied to the Qt's redistributables root folder may be found at

```
C:\Qt\<qt_version>\<compiler_version>\bin
```

while `qwindows.dll`, which is copied to the `platforms` sub-folder must be obtained from

```
C:\Qt\<qt_version>\<compiler_version>\plugins\platforms
```

In these two paths, `<qt_version>` stands for the specific version of Qt installed (such as 6.2.4) and `<compiler_version>` for the version of the compiler for which the pre-built Qt binaries have been installed, such as `msvc19_64`.

Note that the WorkflowMaker's repository already includes the required Qt DLLs. However, these correspond to the version of Qt 6 used when building the applications. If a newer version is used, then these must be replaced by the ones actually used, as described above.

## 3.5  The samples sub-folder

An example of a WorkflowMaker toolkit (as well as workflows and launchers relying on it) has been included. It is a very simple image processing toolkits. The interest of this sample is twofold: first, it makes possible to test the tools in WorkflowMaker without the hurdle of having to define a toolkit from scratch; but, also, they show the architecture of WorkflowMaker-compliant applications. For instance, its tools read the values of their keyboard parameters and input and output file names using an options file, which is implemented using a C++ library named `simple_options_file_parser` included with the samples. The structure of the `samples` sub-folder is shown in Figure 6.
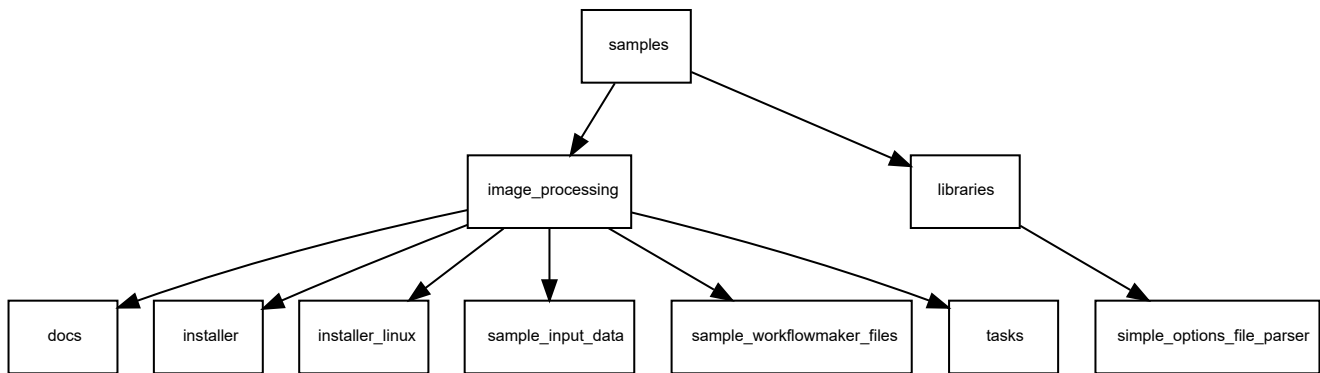


*Figure 6: The structure of the samples sub-folder.*

The library `simple_options_file_parser` is organized in the same way than `WorkflowMakerLib`. Please refer to the description of that library to understand how the sub-folders of `simple_options_file_parser` are structured (in this case, however, there is no `data_to_bin_folder` sub-folder).

The sample image processing toolkit is organized as follows:

- `docs` – The user guides for the toolkit.
- `sample_input_data` – Files that may be used as inputs when running the tools in the toolkit (such

as images).

- `sample_workflowmaker_files` – Example files defining one toolkit, and at least one workflow and launcher relying on the said example toolkit.
- `tasks`- The source code for each of the tools included in the toolkit. These are organized exactly in the same way that the WorkflowMaker tools (ToolkitEditor, WorkflowEditor, WorkflowLauncher) so refer to section 3.6 for details on their structure.

Folders `installer` and `installer_linux` are used to build installers for these samples. Please, refer to the samples documentation for more information about building said samples.

## 3.6 The tools sub-folder

The three tools making WorkflowMaker reside in the `tools` sub-folder. Figure 7 depicts its structure.
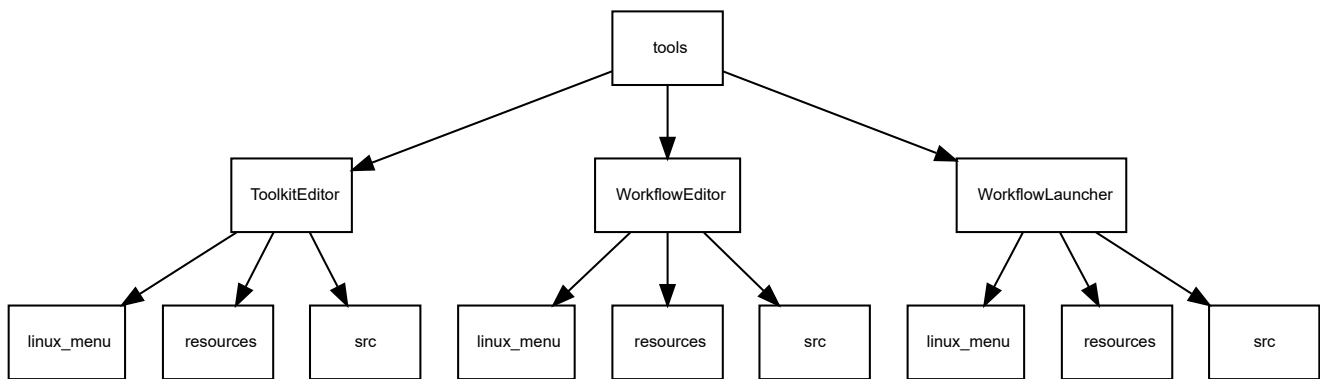


*Figure 7: The structure of the tools sub-folder.*

All tools have been organized in the same way:

- `linux_menu` – Files to build a menu entry for the tool in those Linux distributions that have a system menu.
- `resources` – Images or other files required to build the application or the documentation.
- `src` – The source code.
- 

In the main folder (such as ToolKitEditor) several files are always present:

- *xxxx*.`pro` - This is the Qt project file, used to define the components making the application and giving instructions about how to build it.
- *xxxx*_`resources.qrc` – Definition of the images used by the application. This is a Qt file too.
- `postbuild.bat` & `postbuild.sh` – Shell files for Windows and Linux respectively, stating the steps to execute once the application has been successfully built. These files take care of creating the installer's appropriate sub-folders, copy files there (such as executable or default option files, etc). These constitute the keystone of the build system, since they prepare everything that is needed to build, later, an installer or .deb package for WorkflowMaker.

- *xxxx*.dox – Doxygen documentation file for the tool.

In the bullets above, *xxxx* stands for the name of each of the tools. For instance, the contents of the ToolkitEditor main folder is: `ToolkitEditor.pro`, `ToolkitEditor_resources.qrc`, `ToolkitEditor.dox`, `postbuild.bat` and `postbuild.sh`.

## 3.7 The VC_redistributables sub-folder

This sub-folder is only required for Windows builds.

The VC_redistributables sub-folder contains just a single file, the installer of the Microsoft's Visual C++ redistributables, on which the Windows build of WorkflowMaker relies.

The version provided in the repository is the latest one available at the moment of building the latest version of WorkflowMaker. If new versions of such installer appear (which is related to the appearance of new versions of the  MSVC 2022 compiler) then this file should be replaced by the newer one.

# 4  Building WorkFlowMaker

Building WorkflowMaker means first build the base library (`WorkflowMakerLib`) and then the tools (ToolkitEditor, WorkflowEditor, WorkflowLauncher).

Scripts to **automate** the whole build process for both for the Linux and the Windows platforms have been provided. In all cases, the result of the build process is copied to some directory(ies) so creating an installer or a deb package for WorkflowMaker is simplified, or, as it happens with Windows build, completely automated.

The following sections will discuss how to build WorkflowMaker for Linux and Windows.

⚠️ In these sections, and to simplify the explanations, the WorkflowMaker folder will be referred to as `workflowmaker`. Hence, sub-folders like `data` or `libraries` will be written as `workflowmaker/data` or `workflowmaker/libraries`.

## 4.1  Building for the Linux platform

### 4.1.1 Target platforms

⚠️ As already stated, due to the limited resources available, the build instructions provided in the next sections are guaranteed to work only on modern Ubuntu-based Linux operating systems (such as Ubuntu 24.10 or Linux 22) since version 6 of the Qt framework must be available in the *software repositories* of these operating systems (not downloaded from the Qt website).

### 4.1.2 Are all the required tools available?

WorkflowMaker relies on the Qt framework, more precisely, on Qt version 6. Redistributing applications relying on this framework may be difficult when working on Linux platforms.

As stated above, this guide assumes that Qt6 may be installed on the target computer by means of the system's repositories; this applies to both the regular and *development* Qt6 packages.

Relying on system packages makes the deployment of WorkflowMaker easier just by stating its dependencies in the deb package. Then, if Qt6 is not present when the deb package is installed, the operating system will download and install, automatically, the missing Qt6 libraries.

The list of tools required to develop (and thus build) WorkflowMaker on Ubuntu-based Linux computers are listed in section 2. **Below, there is a list of the commands to type on a *command prompt* to install the complete set of tools:**

- Update the list of packages:
  - `sudo apt update`
- The dpkg and dpkg-dev packages, to be able to build .deb packages and find their

dependencies.
- ○ `sudo apt install dpkg`
- ○ `sudo apt install dpkg-dev`
- gcc / g++
  - ○ `sudo apt install build-essential`
- qt6
  - ○ `sudo apt install qt6-base-dev qt6-tools-dev qt6-tools-dev-tools`
- Qt creator (the Qt IDE, Integrated Development Environment)
  - ○ `sudo apt install qtcreator`
- doxygen
  - ○ `sudo apt install doxygen`
- graphviz
  - ○ `sudo apt install graphviz`

⚠ It is worth to highlight that Qt6 is made of many more libraries than those installed by the command under bullet point "qt6" above. However, these are enough to develop and build WorkflowMaker.

⚠ Note that the version of Qt creator installed as shown above be a little bit outdated, which is not a problem to develop / build WorkflowMaker. If a more recent version is preferred, then it may be obtained directly from Qt's website.

With regard to manual building (see section 4.1.4) and regardless the version of Qt Creator installed, it is necessary to configure it by selecting the "kit" installed in bullet point "qt6". That is, the qt libraries used by this IDE must be provided by the system repositories, not downloaded from the Qt website. Figure 8 depicts Qt Creator's `Preferences/Kits` windows. There, highlighted in red, it may be seen how the "system" Qt libraries have been selected. The path to the `qmake` tool is also highlighted, which will be needed to adapt the automated build scripts when building WorkflowMaker (see section 4.1.3).
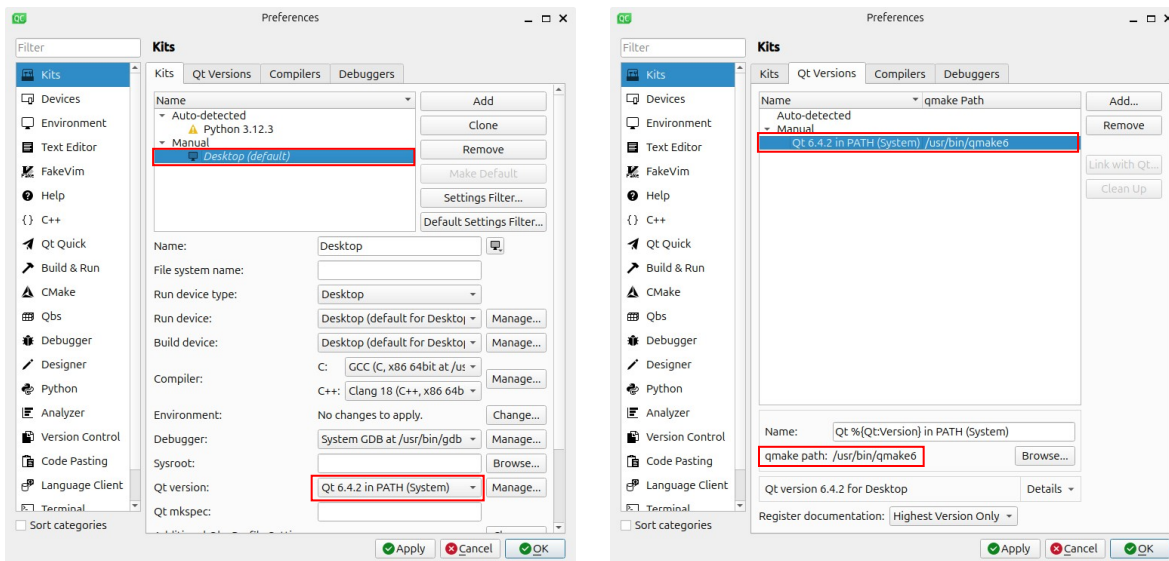


*Figure 8: Selecting the right Qt libraries for Qt creator. The path to qmake is shown.*

## 4.1.3 Automated building

The following are the step-by-step instructions to build WorkflowMaker using the automated script.

**First of all, take care of changing the version string of the software,** but **only** if changes have made; **this is not necessary when building the code just downloaded from the repository.** **Version changes must be reported in the following files:**

- `workflowmaker/libraries/WorkflowMakerLib/data_to_bin_folder/workflowmaker_version.txt` – The typical version string follows the pattern x.y.z (e.g., 1.0.8). Change this value if necessary and save the file.
- `workflowmaker/installer/installer_sw_linux/control` – Change the line "`Version: x.y-z`" to match the actual version. **Note the dot-dash pattern** between x and y and then between y and z.

**Open a terminal. Change the default directory** to `workflowmaker`

```
cd  workflowmaker
```

**Check that the build script (file build_all.sh in this folder) points to the right `qmake` tool.** To ascertain what is the path to `qmake`, open Qt Creator, Then go to `Edit→Preferences`, and select the `Kits` tab in the vertical tab list, and then `Qt Versions` in the horizontal. There (see the lower right part of Figure 8) the path to `qmake` is shown highlighted in red. Figure 9 below shows the first lines of `build_all.sh`; the line to check and eventually change has been highlighted.
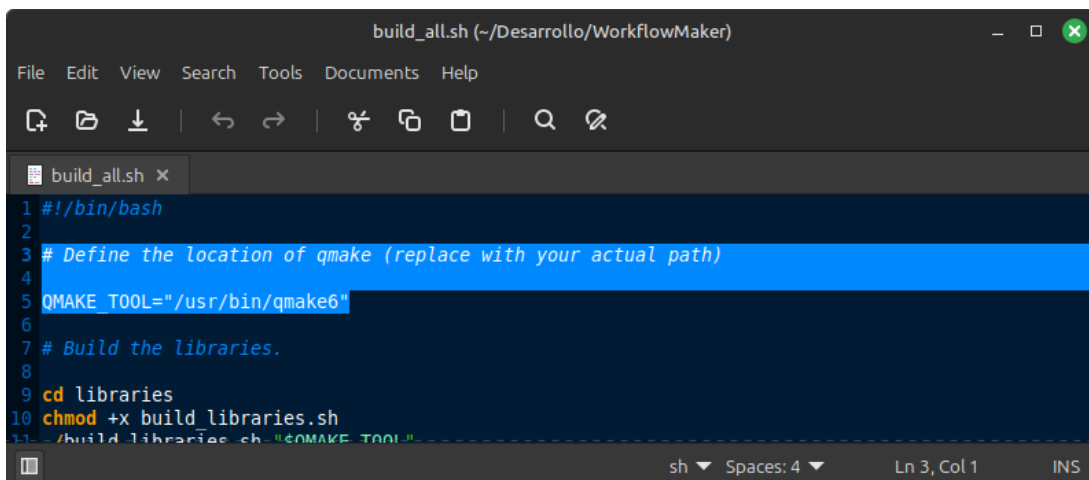


*Figure 9: Setting the path to the qmake tool.*

The line `QMAKE_TOOL="/usr/bin/qmake6"` states the path to the `qmake` tool. Note that this value (`/usr/bin/qmake6`) is obtained from the path shown in Figure 8.

If the value of the line above does not match the one shown in Figure 8, change it accordingly and save the `build_all.sh` file. If these values are the same, just close the file.

**Make sure that the file build_all.sh has the execution privilege:**

```
chmod +x build_all.sh
```

**Run the build script:**

```
./build_all.sh
```

The result of the execution of the script is that a new folder structure has been created and populated with all the necessary files to build the deb package (see section 4.1.5). Moreover, another folder, named `workflowmaker_x.y-zz_doc_and_samples` is created. It is stored in `workflowmaker/installer/installer_sw_linux` and contains the documentation (user guides) as well as some sample files for the image processing sample toolkit. Its structure is so simple that it is not necessary describe it here.

## 4.1.4 Building by hand

Alternatively, it is possible to build WorkflowMaker by hand, building, one by one, the several projects if is composed of. However, when building WorkflowMaker *by the first time* (that is, when it is downloaded from the repository) the automated build process is recommended.  Later on, when changes are made to specific components of WorkflowMaker (such as a single tool, or the library), building by hand makes much more sense.

The general process to build a single library or a tool is as follows:

- Open Qt Creator.
- Select the `.pro` file for the desired target (the library or any of the tools).
- Configure the project (selecting the Qt 6 installed in your computer and the target output).
- Build.

⚠ Building manually any of the components, either library or tools, will also copy the corresponding files to the deb folder structure (see section 4.1.5) as it is done by the automated process.

It is worth noting that changing the WorkflowMakerLib library implies rebuilding all the tools to guarantee that these changes are conveniently propagated. In general, when building manually, libraries must be built first, then the tools.

## 4.1.5 The deb folder structure

The WorkflowMaker build script creates a folder structure holding the necessary files to, later, create a deb package. This folder is always named **`workflowmaker_x.y-zz_amd64`** and it is placed in the sub-folder `workflowmaker/installer/installer_sw_linux`, that is, its full path is

```
workflowmaker/installer/installer_sw_linux/workflowmaker_x.y-zz_amd64
```

Each element built (the library, each tool) copies one or more elements to this folder structure. When everything is built, the deb package may be created (see sections 4.1.6 to 4.1.8). Figure 10 on page 17 depicts the structure of the said folder as created by the build process. The contents of each sub-folder is described below.

- Root folder (that is, `workflowmaker_x.y-zz_amd64` itself). A file named `find_dependencies.sh` is copied here. It may be used to ascertain what are the dependencies (in terms of system libraries) on which the tools in WorkflowMaker rely. *When said dependencies are already known, this file must be deleted.*
- `DEBIAN` – It contains the `control`, `postinst` and `prerm` files.
  - `control` – Defines the contents of the package, according to the syntax required for this files. It includes the name of the package, its version, the packages on which it depends, for instance. If the software is changed, this file must be modified in order to reflect any changes (such as, of course, the version number or dependencies).
  - `postinst` – This is a shell file that is executed as the last step when the deb package is installed. In the case of WorkflowMaker, it takes care of creating the menu entries for the different tools.
  - `prerm` – It is executed as the first step when the deb package is uninstalled. In this case it takes care of removing the menu entries for the applications.
  - All these files are copied from `workflowmaker/installer/installer_sw_linux`.
- `debian` – This folder contains an extra copy of the `control` file (see `DEBIAN` sub-folder above) since, when finding dependencies, the existence of this sub-folder (`debian`) as well as said `control` file are required. *Note that this sub-folder must be deleted once the dependencies have been ascertained, prior to building the deb package. This is why it is highlighted in red in Figure 10.*
- `usr/local/bin` – Here, the executable files for the tools as well as the XML schema definitions for toolkits, workflows and launchers are copied. The file with the package version string is also stored here. The executable files are produced by building the respective tools; the XML schema definitions are copied from `workflowmaker/data`. The version string file is copied from `workflowmaker/libraries/WorkflowMakerLib/data_to_bin_folder`.
- `usr/share/workflowmaker` – Here, the .desktop and icons for the tools are stored. These are copied, for each tool, from their `linux_menu` subfolders; for instance, the files for ToolkitEditor are copied from `workflowmaker/tools/ToolkitEditor/linux_menu`.

## 4.1.6 Before building the deb package: finding dependencies

⚠️ This step may be skipped most of the times since dependencies do not change so often. In fact, **it may be ignored when building WorkflowMaker just after downloading it from the repository.**

Finding the dependencies of WorkflowMaker is something that must be done every now and then, since the contents of the system's (operating system) repositories may vary, including newer versions of some libraries needed by the tools.
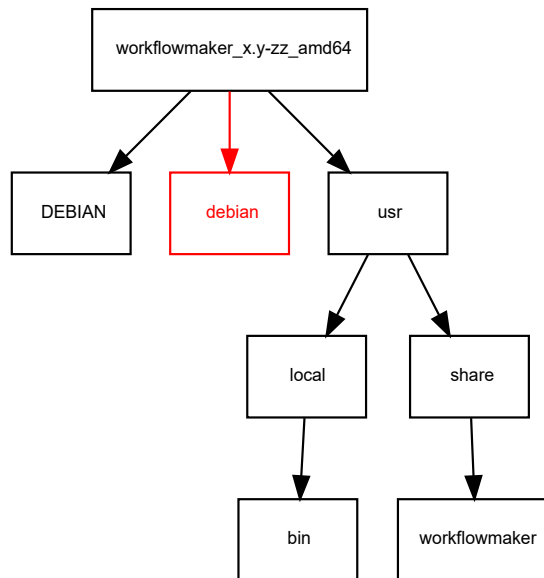
*Figure 10: The structure of the .deb folder as created by the build process.*

The `control` file stored in the `debian` and `DEBIAN` directories (see Figure 10) contain the dependencies at the moment of writing this guide. If you think that it is convenient to ascertain if these dependencies have changed, proceed as follows:

- Edit the `control` file in the `debian` (not `DEBIAN`) folder.
- Add an extra line, just under "`Package: workflowmaker`" saying "`Source: workflowmaker`". See the highlighted lines in Figure 11. Ignore, by now, the "Dependencies:" field in this file, since it states the old dependencies for WorkflowMaker.
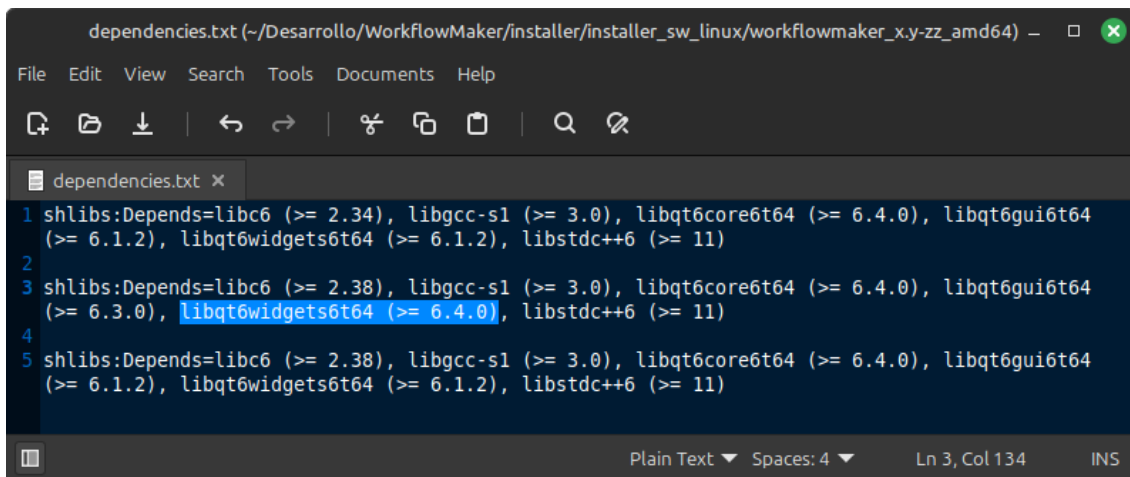- Save the file.



*Figure 11: Modifying the control file to find dependencies.*

Then,

- Open a command prompt.
- Change the default directory to `workflowmaker_x.y-zz_amd64`.
- Type the following command: `./find_dependencies.sh`.

The command above produces an output file, named `dependencies.txt`. There, the dependencies for all the tools in WorkflowMaker are listed. See Figure 12. There, a line is included for each tool. Note that different versions of a system library may be required: as seen in said figure, versions 6.1.2 and 6.4.0 of libqt6widgets64 are required by different tools. In these cases, the highest version must be used.



*Figure 12: The output of find_dependencies.sh.*

The idea is to obtain the unique list of dependencies (keeping the highest versions) written in the format required by the control file. In the case of the example above, this line would be:
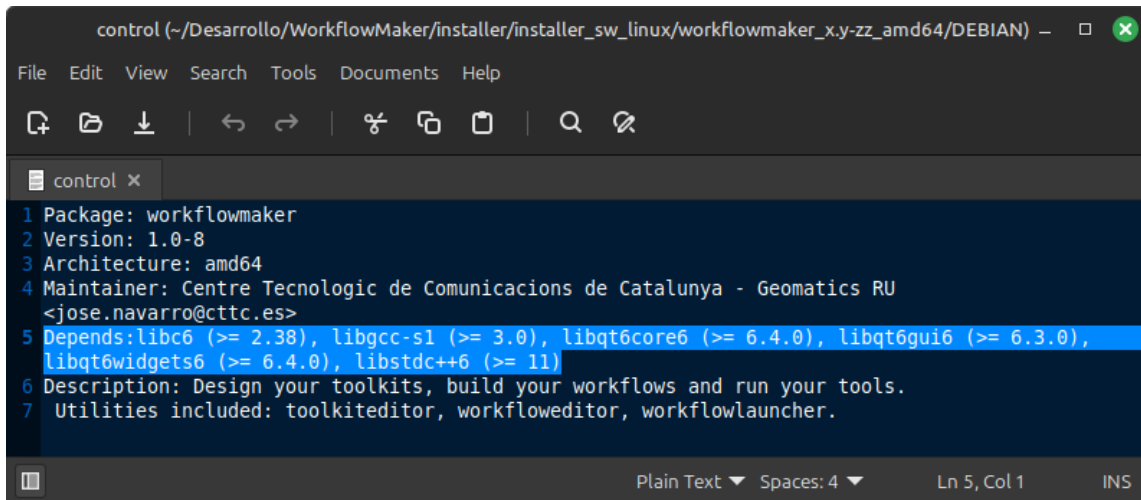
```
Depends:libc6 (>= 2.38), libgcc-s1 (>= 3.0), libqt6core6 (>= 6.4.0), libqt6gui6 (>= 6.3.0),
libqt6widgets6 (>= 6.4.0), libstdc++6 (>= 11)
```

Note the use of the highest version required for each library in the dependencies set.

Then, **delete** the file `dependencies.txt` created by `find_dependencies.sh` in the `debian` folder since it is no longer needed. Delete `find_dependencies.sh` as well.

The next step is to **edit the `control`** file located in the `DEBIAN` (not `debian`) folder and replace the line with the dependencies with the ones just found. Figure 13 depicts this change (see the highlighted line).

Finally, **save the `control`** file.

*Figure 13: Updating the DEBIAN/control file with the new dependencies.*

### 4.1.7 Purge the deb folder

As stated in section 4.1.5 the build process (either automated or manual) creates the folder named `workflowmaker_x.y-zz_amd64` where the several files making the deb package are stored. This folder included several sub-folders, among which one named `debian` (not `DEBIAN`) could be found. Figure 10 shows in red said folder. The purpose of this `debian` folder was to make possible finding the dependencies of the tools in WorkflowMaker (section 4.1.6).

**At this point of the process, the `debian` folder must be deleted**, since it may not be part of the deb folder structure when the deb package is built. Therefore, it must be removed. Figure 14 shows the structure of `workflowmaker_x.y-zz_amd64` after this removal takes place. Compare it with Figure 10.

Now, it is possible to build the deb package.

### 4.1.8 Creating the deb package

Once that the library and the tools have been built, either automatically or manually, and the deb package folder purged, it is possible to create a deb package. To do it, follow the steps below.

**Rename** the folder `workflowmaker_x.y-zz_amd64` located in `workflowmaker/installer` and name it according to the version it implements. For instance, if the new version is 1.0.8, then the new name should be `workflowmaker_1.0-8_amd64`.

**Open a terminal. Change the default directory** to `workflowmaker/installer/installer_sw_linux`

```
cd  workflowmaker/installer/installer_sw_linux
```
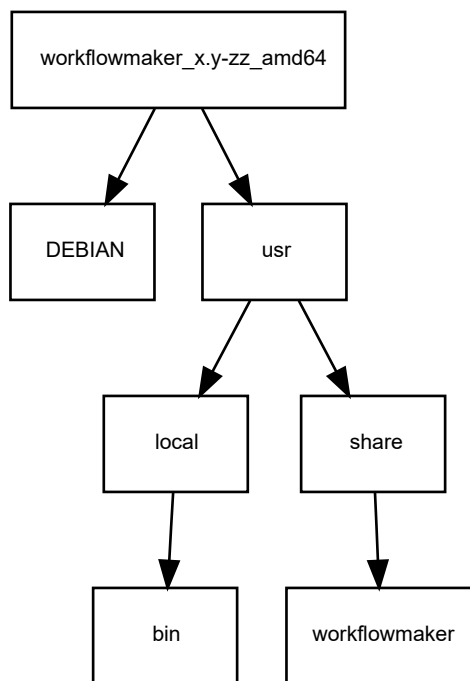
*Figure 14: The purged deb package folder.*

**Run the following command:**

```
dpkg-deb --build --root-owner-group <deb-folder>
```

where `<deb-folder>` stands for the deb folder just renamed. For instance, if the original `workflowmaker_x.y-zz_`folder has been renamed to `workflowmaker_1.0-8_amd64`, then the command should be:

```
dpkg-deb --build —root-owner-group workflowmaker_1.0-8_amd64
```

Obviously, the name of the `<deb-folder>` must be changed in the example above to match the actual name of the deb folder.

This will create the deb file, which may be used to install WorkflowMaker in Debian-based Linux distributions. **It is created in the `workflowmaker/installer/installer_sw_linux` folder.**

The name of the deb package will be `workflowmaker_x.y-z_amd64.deb`, where `x.y-z` stands for the actual version built. If it is, for instance, 1.0-8, then the name of the package will be `workflowmaker_1.0-8_amd64.deb.`

## 4.2  Building for Windows platforms

### 4.2.1 The tools

The tools required to develop / build WorkflowMaker are listed in section 2. Installing these is straightforward thanks to the GUI-based installers provided.

It is worth to highlight, however, that Qt 6 must be installed *using the official Windows x64 online installer* available at the Qt website (https://www.qt.io/download-open-source). When installing, take care of including the "Developer and Designer Tools" as well as the Qt 6 libraries (see Figure 15).



*Figure 15: Installing Qt 6 on Windows - Component selection.*

The Qt version selected in the figure above (6.2.4) was the one closest to that available on Linux at the moment of writing this guide. Later versions (such as 6.8.0, at the top of the bigger red rectangle) may be selected. However, versions greater than 6.2.4 have not been tested (although no problems should be expected).

Note that the libraries for the MSVC compiler (underlined in the figure) have been included. These are the equivalents of the Linux Qt system libraries (see section 4.1.2).

## 4.2.2 The structure of the windows installer folder

When building WorkflowMaker for Windows, the build scripts create automatically a folder where all the files required to make an installer will be copied. This folder, named `deployment`, is created inside the Windows installer sub-directory, that is, at

`workflowmaker/installer/installer_sw_windows/deployment`

Figure 16 depicts the structure of said `deployment` folder. These are the contents copied there by the build scripts:

- `deployment/bin`
  - The executable files of the several tools (created by the build process).
  - The XML schema files defining the toolkit, workflow and launcher (`toolkit.xsd, workflow.xsd, launcher.xsd`). Copied by the build process from folder `workflowmaker/data`.
  - The Qt redistributables. Copied from `workflowmaker/QT_redistributables`.
  - The MSVC redistributables installer. Copied from `workflowmaker/VC_redistributables`.
  - The version string file, `workflowmaker_version.txt`, copied from `workflowmaker/libraries/WorkflowMakerLib/data_to_bin_folder`.
- `deployment/data_samples`. A set of files consisting of a toolkit for image processing, a workflow example relying on this toolkit, and an example laucher. All these files are copied from `workflowmaker/samples/image_processing/sample_workflowmaker_files`.
- `deployment/docs` .The documentation folder. All PDFs are copied from `workflowmaker/docs/user guide`.
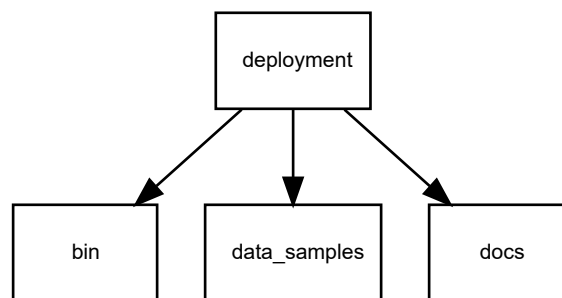


*Figure 16: The structure of the Windows (installer) deployment folder.*

## 4.2.3 Automated building

This is the recommended procedure to build WorkflowMaker just after its source code has been downloaded from the repository. The following are the step-by-step instructions to build WorkflowMaker using the automated script.

**First of all, take care of changing the version string of the software, but only if changes have made; this is not necessary when building the code just downloaded from the repository**. Version changes must be reported in two files.

The first one is:

```
workflowmaker/libraries/WorkflowMakerLib/data_to_bin_folder/workflowmaker_version.txt
```

The typical version string follows the pattern x.y.z (e.g., 1.0.8). Change this value if necessary in the unique line this file contains and then save it.

The second file to change is the NSIS script controlling how the installer is built. This file is located at:

```
workflowmaker/installer/installer_sw_windows/WorkflowMaker.nsi
```

Edit the NSIS script and look for a line similar to:

```
!define defVersion        "1.0.7"
```

at the beginning of the file. Change the version string accordingly and save the file. For instance, if the new version is 1.0.8, the new version string should be (in red, the changes):

```
!define defVersion        "1.0.8"
```

**Open a MSVC 2022 x64 terminal.**

⚠️ Beware that a *regular* Windows terminal is not enough to build WorkflowMaker. Instead, a specific MSVC 2022 terminal to build x64 applications must be used. These may be started from the Visual Studio menu (in the Windows start menu). Its precise name is "*x64 Native Tools Command Prompt for VS 2022*".

**Change the default directory** to `workflowmaker`

```
cd  workflowmaker
```

**Run the build script:**

```
build_all.bat
```

The result of the execution of the script is that the deployment folder structure has been created and populated; *the installer is built as well.* It is located in folder:

```
workflowmaker/installer/installer_sw_windows
```

and its name is

```
WorkflowMaker-x.y.z-setup.exe
```

where *x.y.z* stands for the version of the package. Thus, if for example the version string is 1.0.8, then the actual name of the installer would be:

<div align="center">

`WorkflowMaker-x.y.z-setup.exe`

</div>

## 4.2.4 Building by hand

Alternatively, it is possible to build WorkflowMaker by hand, building, one by one, the several projects if is composed of. However, when building WorkflowMaker *by the first time* (that is, when it is downloaded from the repository) the automated build process is recommended.  Later on, when changes are made to specific components of WorkflowMaker (such as a single tool, or the library), building by hand makes much more sense.

The general process to build a single library or a tool is as follows:

- Open Qt Creator.
- Select the `.pro` file for the desired target (the library or any of the tools).
- Configure the project (selecting the Qt 6 installed in your computer and the target output).
- Build.

⚠️ Building manually any of the components, either library or tools, will also copy the corresponding files to the `deployment` folder structure (see section 4.2.2) as it is done by the automated process.

It is worth noting that changing the WorkflowMaker library implies rebuilding all the tools to guarantee that these changes are conveniently propagated. In general, when building manually, libraries must be built first, then the tools.

## 4.2.5 Building the installer (when building manually)

When changes are made to individual components of WorkflowMaker and these changes are built using the manual procedure (section 4.2.4) the installer is not rebuilt automatically. In these cases, it is necessary to re-build it by hand.

⚠️ Before rebuilding the installer by hand, make sure that *all* the tools that have been changed and therefore included in the new version of the installer have been built.

**Change the version string, but only if changes have made; this is not necessary when building the code just downloaded from the repository.** Version changes must be reported in the file:

<div align="center">

`workflowmaker/libraries/WorkflowMakerLib/data_to_bin_folder/workflowmaker_version.txt`

</div>

The typical version string follows the pattern x.y.z (e.g., 1.0.8). Change this value if necessary in the unique line this file contains and then save it.

**Open the installer script.**

This is file `workflowmaker/installer/installer_sw/WorkflowMaker.nsi`.

Open this file using the HM NIS EDIT editor (see section 2) **only**, since it is integrated with the NSIS installer system and only from here it will be possible to build the installer.

**Change the version of the installer, but again, only if changes have been made.**

Do this only when the version of the installer must reflect a change in the version of the software. If the installer is just being regenerated, it is not necessary to change the installer's version.

To change the version, look for the following line close to the beginning of `WorkflowMaker.nsi`:

```
!define defVersion "1.0.7"
```

(the actual version number may be different) and change the version string as needed. For instance:

```
!define defVersion "1.0.8"
```

**Save the installer script. Only** if it has been modified by the previous step.

**Build the installer.**

To do it, click on the build button in the editor's toolbar (see Figure 17).

**Close HM NIS Edit.**

The installer is created in the folder

```
workflowmaker/installer/installer_sw
```

and it is named

```
WorkflowMaker-x.y.z-setup.exe
```

where x.y.z corresponds to the version string indicated in the installer script (see previous steps). Thus, if such version string is, for instance, 1.0.8, the name of the installer woud be:

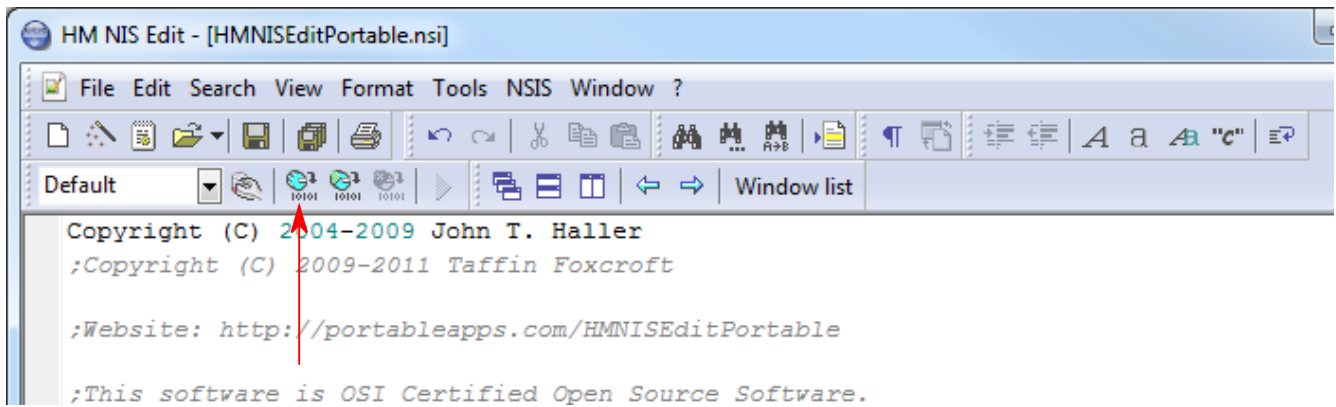```
WorkflowMaker-1.0.8-setup.exe
```

*Figure 17: Building the windows installer using HM NIS Edit.*

Running this installer will install WorkflowMaker on the target Windows computer. Note that it includes not only the applications but also the documentation and samples.

## 4.3  The API documentation

When the source code for WorkflowMaker is downloaded from the repository, the documentation for the APIs of the libraries or tools is included.

Note, however, that *this documentation is regenerated automatically each time WorkflowMaker is rebuilt  by means of the automated script (sections 4.1.3 and 4.2.3) or whenever any of its components (library, tools) are built manually (sections 4.1.4 and 4.2.4).* In this way, and assuming that the documentation is properly updated in the source code when changes are made, the API documentation will always be up to date. This task is assumed by the postbuild.bat / postbuild.sh and .dox files included with each library or tool.

The documentation is created in a new folder named `doc_html`, which is located in the respective directories of each of the WorkflowMaker libraries or tools. As the name of the folder suggests, the documentation is created in HTML format; the main page of the documentation (for each library of tool) is the file `index.html`.

Figure 18 depicts, highlighted in red, the location of this folder for the ToolkitEditor tool.
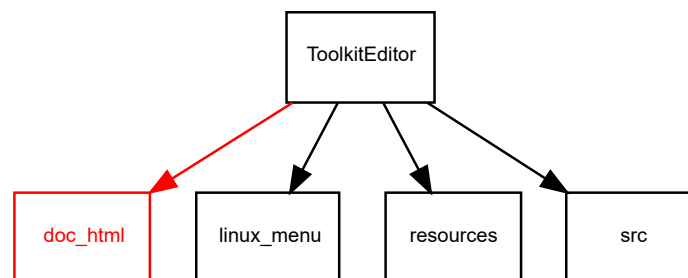


*Figure 18: The doc_html folder for the ToolkitEditor tool including its API documentation.*

Consequently, to open the API documentation for each of the components making WorkflowMaker, the files to check are, respectively:

- Libraries
    - WorkflowMakerLib
        `workflowmaker/libraries/WorkflowMakerLib/doc_html/index.html`
    - rapidxml
        There is no documentation folder for this library. It is an open source project that may be found here:
        [https://rapidxml.sourceforge.net/](https://rapidxml.sourceforge.net/)
        The documentation for this library may be checked here:
        [https://rapidxml.sourceforge.net/manual.html](https://rapidxml.sourceforge.net/manual.html)
- Tools
    - ToolkitEditor
        `workflowmaker/tools/ToolkitEditor/doc_html/index.html`
    - WorkflowEditor
        `workflowmaker/tools/WorkflowEditor/doc_html/index.html`
    - WorkflowLauncher
        `/home/janavaro/Desarrollo/WorkflowMaker/tools/WorkflowLauncher/doc_html/index.html`

As stated above, these files will not be available until WorkflowMaker is built at least once.

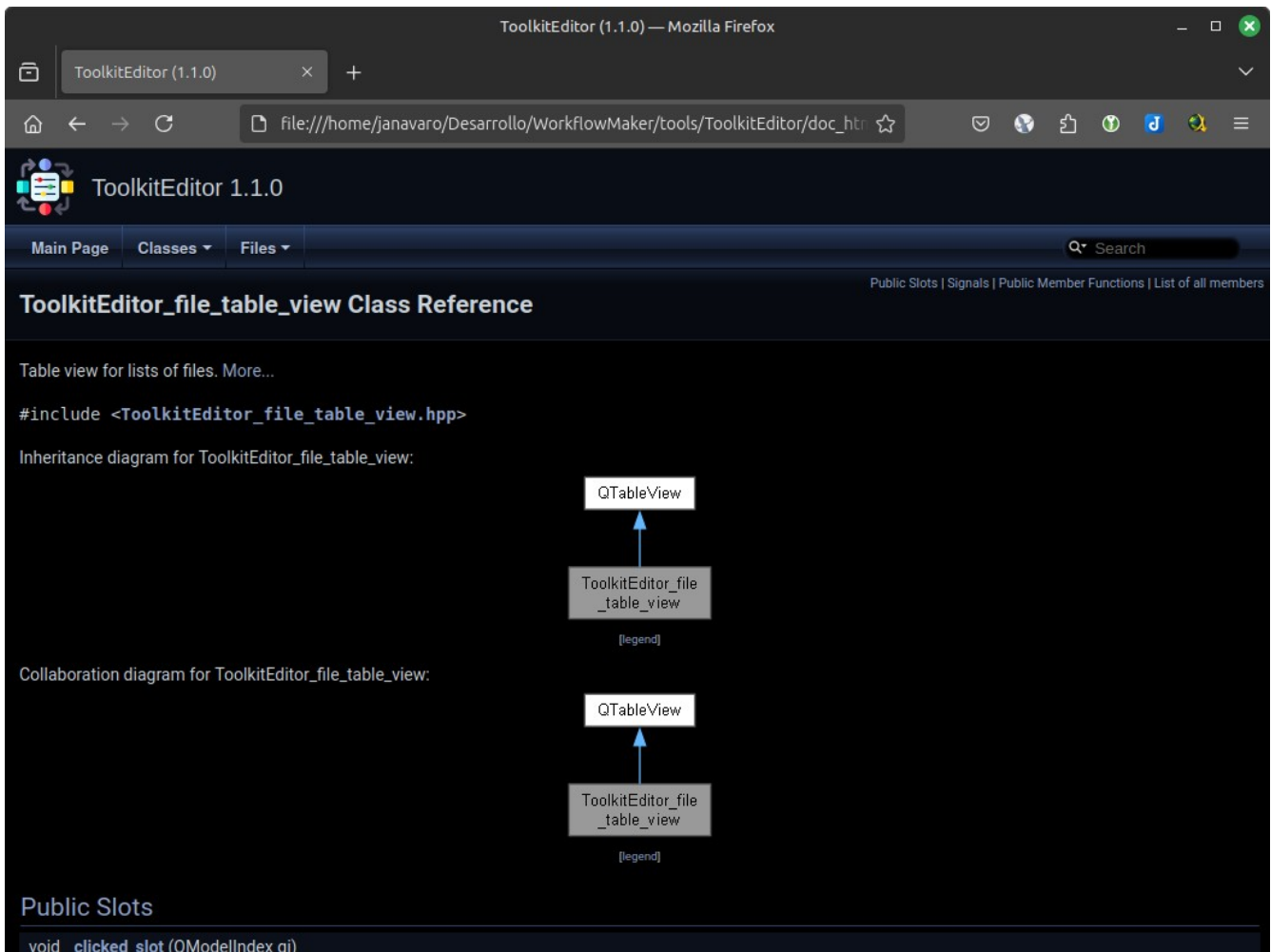Figure 19 shows the documentation of one of the classes included in the ToolkitEditor tool.

*Figure 19: An example of the API documentation in HTML format.*

# 5   Open Source licenses

WorkflowMaker relies on several open source libraries and it is also distributed as open source.

## 5.1   The rapidxml library

This application uses the rapidxml library, which is available under the Boost Software License. For more information, visit https://www.boost.org/LICENSE_1_0.txt.  A copy of this license may be found in the `workflowmaker/doc/licenses` folder.

## 5.2   The Qt 6 framework

This application uses the Qt 6 framework, which is available under the LGPL v3 license. For more information, visit https://www.qt.io/licensing and https://www.gnu.org/licenses/lgpl-3.0.html. A copy of the LGPL v3 license may be found in the `workflowmaker/doc/licenses` folder.

## 5.3 Images

All the icons used by the applications have been built using images downloaded from freepik.com.
These have been the images used:

adaptation_5266710.png
filter_3126489.png
gear_937458.png
pencil_10183063.png
rocket_8661082.png

## 5.4 WorkflowMaker

This project is licensed under the MIT License. See the LICENSE file in the root folder of the source
code distribution for details.