

# MDE4CPP

## Setup & Installation Guide

For Windows



## Table of contents

<b>1. Introduction .....</b>	<b>2</b>
1.1. Required Software .....	2
<b>2. Installation Process .....</b>	<b>3</b>
2.1. Setting up Java Development Kit .....	3
2.2. Setting up Eclipse Modelling Framework .....	4
2.3. Setting up MinGW-Compiler .....	12
2.4. Setting up Gradle .....	14
2.5. Setting up CMake .....	16
<b>3. Getting MDE4CPP .....</b>	<b>18</b>
3.1. Setting up Git .....	18
3.2. Cloning the Repository .....	19
<b>4. Setting up MDE4CPP .....</b>	<b>21</b>
4.1. Environment Configuration .....	21
4.2. Building the Project .....	23

# 1. Introduction

This Installation Guide provides detailed step-by-step instructions on how to set up all the required software for proper usage of the MDE4CPP project as well as setting up and configuring MDE4CPP for first use.

Section 1.1 lists all required software that you will have to install and set up in order to be able to set up and work with MDE4CPP.

Section 2 provides step-by-step instructions on how to set up (and configure if applicable) all the required software mentioned in section 1.1 (including the Eclipse Modelling Framework and required Plugins).

There are two possible ways of how to get MDE4CPP (all project files and directories) onto your local machine, which are explained in section 3 (along with setting up possibly required additional software).

Section 4 provides instructions on how to configure required environment variables for the build process, as well as on how to build the project itself using the Gradle software, once it was brought to your local machine.

For further information on the MDE4CPP project please see the [project site](#).

## 1.1. Required Software

The following list names all the software required to set up and work with MDE4CPP on your local machine.

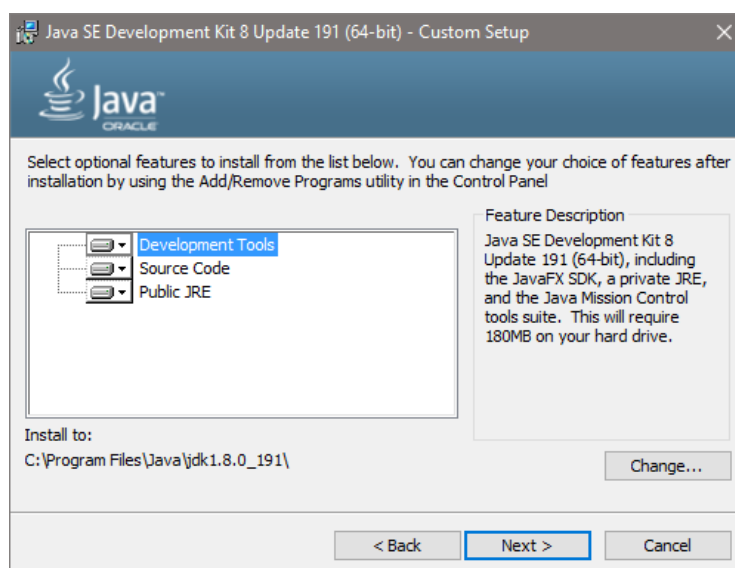
Software	Recommended Version
Eclipse Modelling Framework (including required plugins)	2018-09, Photon, Oxygen or older version
Java Development Kit (JDK)	version 1.8
MinGW-C++-Compiler	64-bit version required
Gradle	no specific recommendation (preferably newest)
CMake	no specific recommendation (preferably newest)

## 2. Installation Process

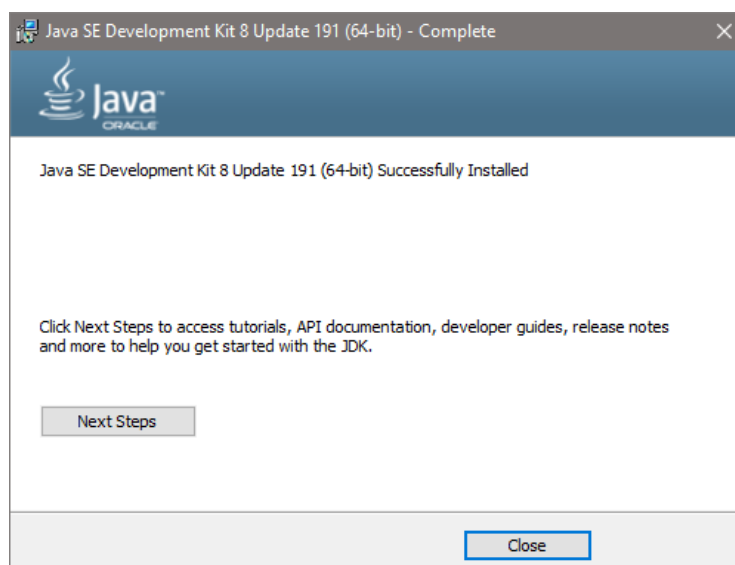
### 2.1. Setting up Java Development Kit

**Step 1:** Download the installation file for *Java Development Kit 1.8*. Choose an installation file suitable for your system [here](#).

**Step 2:** Start the Installer. After clicking **Next** on the initial dialog, choose the components that you want to install. For this guide, we install the full package and leave the options as is. Next, choose your installation directory. For this guide, we choose **C:\Program Files** as installation directory. Click **Next**.



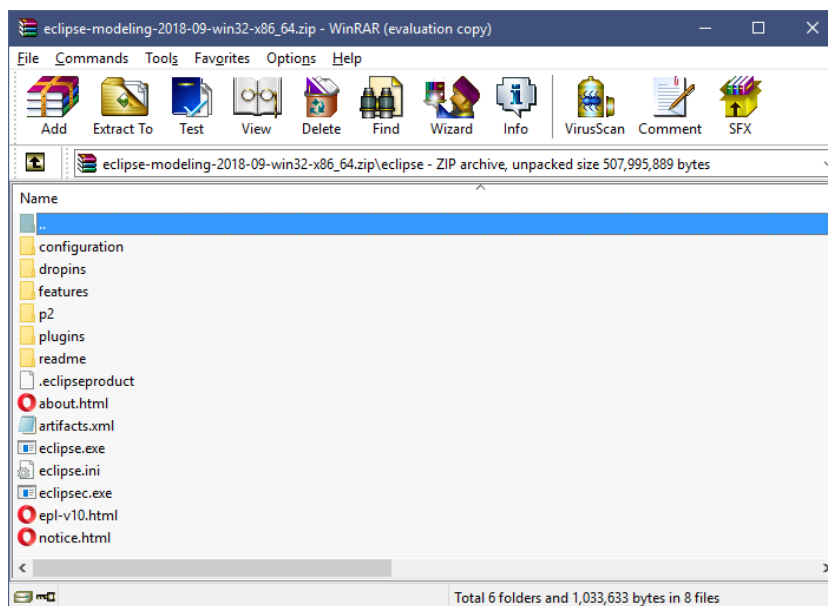
**Step 3:** After the installation process has finished, click **Close** to finish the setup or **Next Steps** for further information about *JDK1.8*.



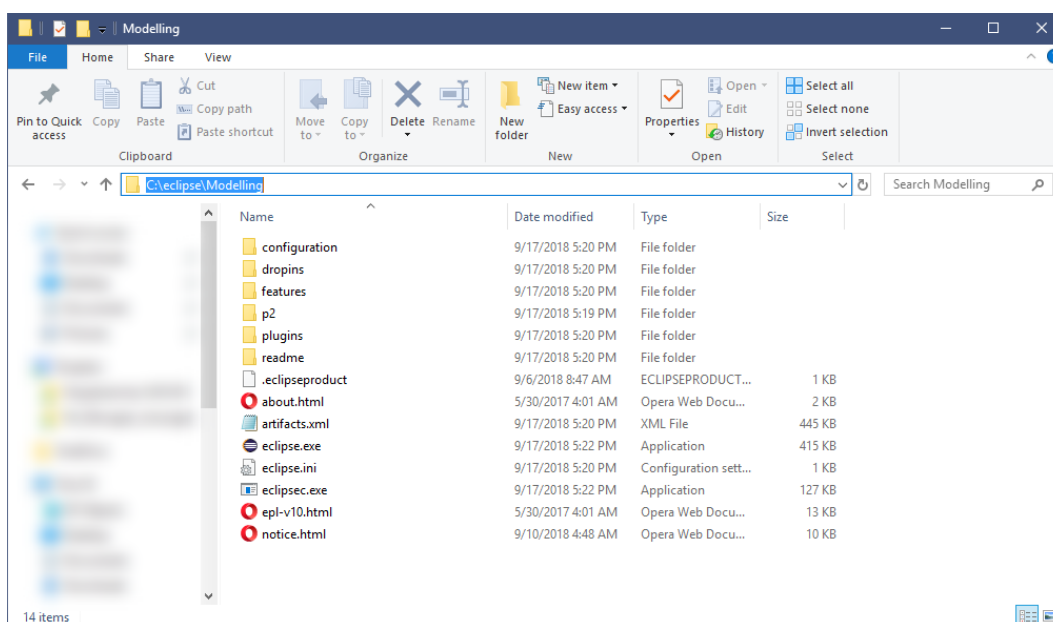
## 2.2. Setting up Eclipse Modelling Framework

### 2.2.1. Setting up Eclipse Modelling Tools

**Step 1:** Download an out-of-the-box package of *Eclipse Modelling Tools*. You can download a version suitable for your system [here](#) (scroll down and search for **Eclipse Modelling Tools**). For this guide we choose the 64-bit version of the latest release of **Eclipse 2018-09**. After downloading *Eclipse Modelling Tools*, open the archive (e.g. using *WinRAR*).



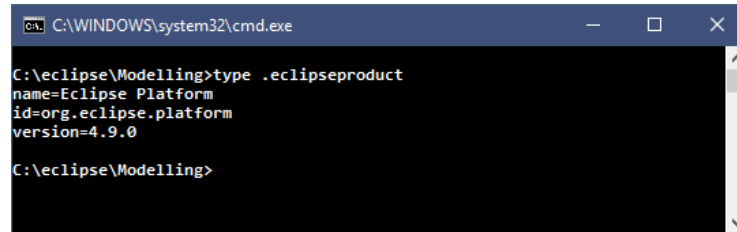
**Step 2:** Choose a directory to extract *Eclipse Modelling Tools*. It is recommended to extract *Eclipse Modelling Tools* to **C:\eclipseModelling** (which we choose for this guide). Create the directory and extract the files.



**Hint:** To check out your version of *Eclipse*, open a command prompt and navigate to your *Eclipse Modelling Tools* directory. Type in the following commands:

```
cd C:\eclipse\Modelling
```

```
type .eclipseproduct
```

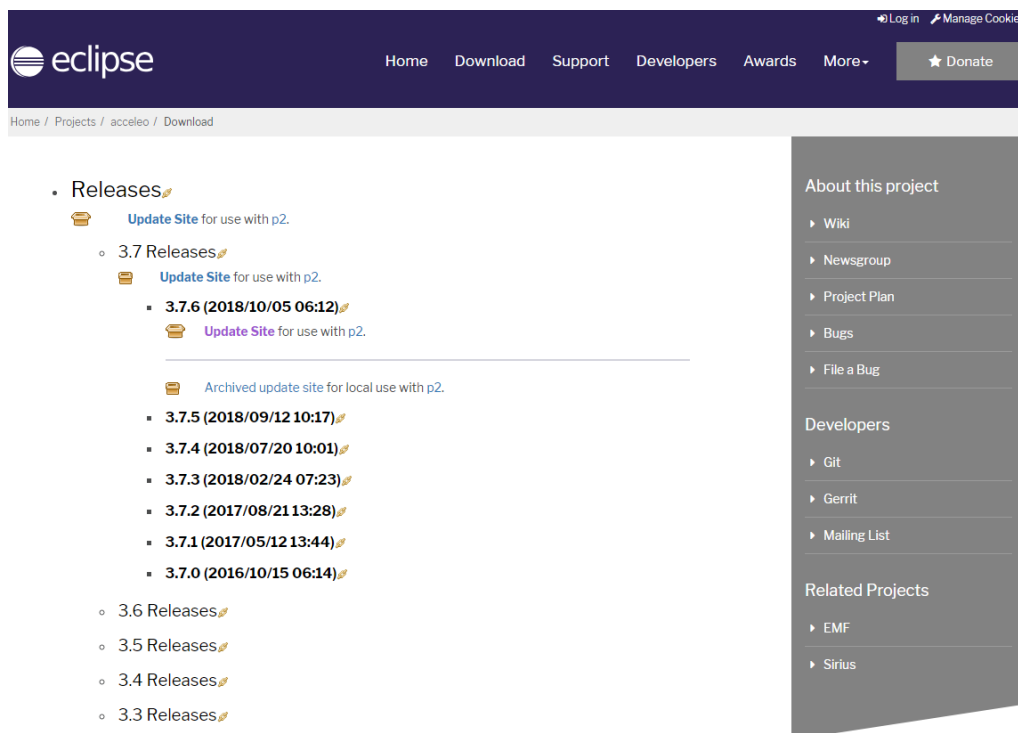


```
C:\WINDOWS\system32\cmd.exe
C:\eclipse\Modelling>type .eclipseproduct
name=Eclipse Platform
id=org.eclipse.platform
version=4.9.0
C:\eclipse\Modelling>
```

**Step 3:** Open ***eclipse.exe*** in your *Eclipse Modelling Tools* directory. You will be asked to define a ***workspace directory***. In this directory, all your future *Eclipse*-projects will be saved (working directory), so choose a directory that you have read and write permission for. You can then choose to set this directory as default or not.

## 2.2.2. Setting up Acceleo

**Step 1:** First you will need a link (URL) to an *Eclipse* software repository. You can find all releases of *Acceleo* [here](#). Choose a version of *Acceleo* and follow the link to the update site.

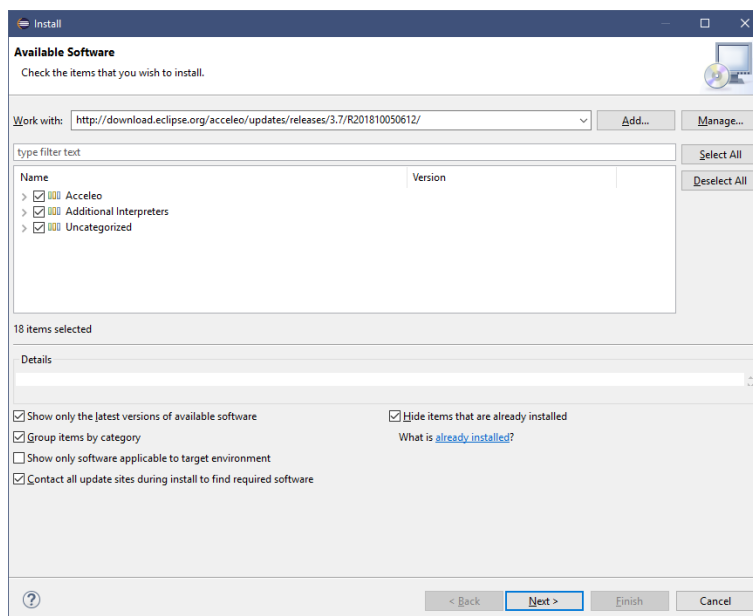


The screenshot shows the Eclipse website's Acceleo download page. The main content area lists releases of Acceleo, with the most recent being 3.7.6 (2018/10/05 06:12). Each release includes a link to an 'Update Site for use with p2'. The sidebar on the right contains links to 'About this project' (Wiki, Newsgroup, Project Plan, Bugs, File a Bug), 'Developers' (Git, Gerrit, Mailing List), and 'Related Projects' (EMF, Sirius).

**Step 2:** Copy the URL of the update site from your browser. Depending on the version of *Acceleo* that you would like to install, this URL could look like this:

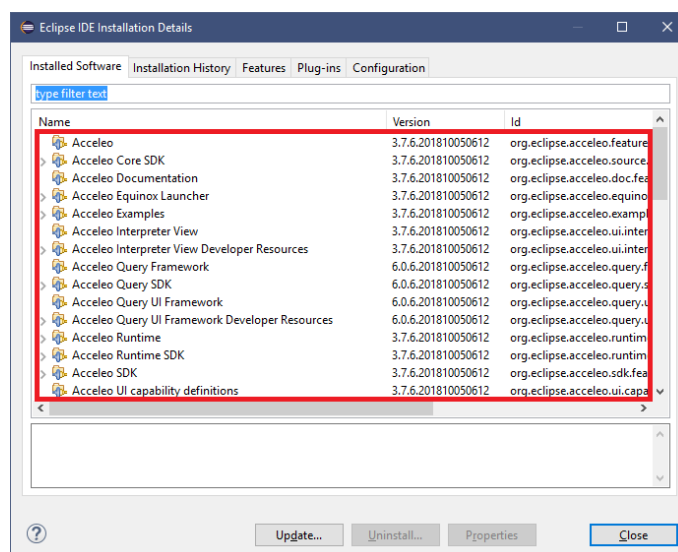
*<http://download.eclipse.org/acceleo/updates/releases/3.7/R201810050612/>*

Open *Eclipse*. From the menu bar, choose **Help** → **Install New Software**. Paste the URL of the update site into the “**Work with**”-input-box and click **Add...** → **Add**.



**Step 3:** Check all three categories to install *Acceleo* completely (as shown above). Click **Next**. *Eclipse* will now check requirements and dependencies. After that, click **Next** → **Finish**. *Eclipse* will now install all software components of *Acceleo* in the background. When the installation process has finished, you will have to restart *Eclipse*.

To check if the installation was successful, open *Eclipse* and navigate to **Help** → **About Eclipse IDE** → **Installation Details** → **Installed Software**.





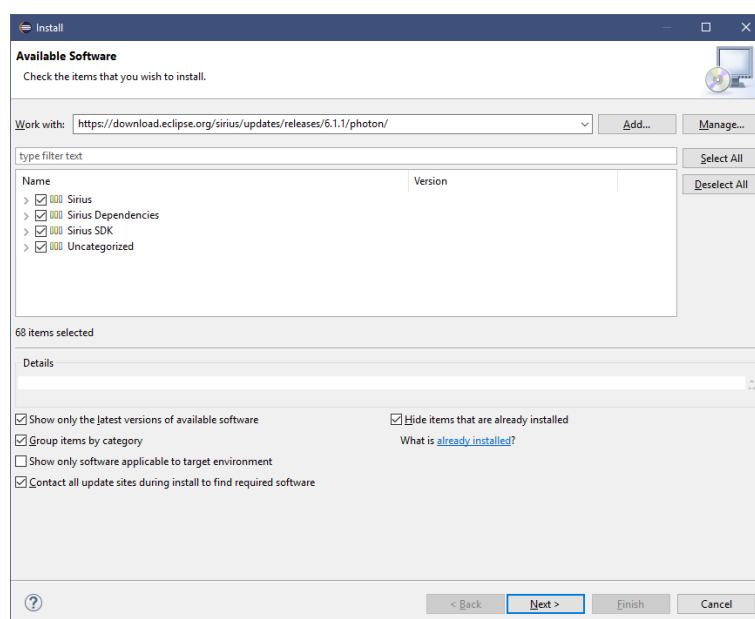
### 2.2.3. Setting up Sirius

**Step 1:** First you will need a link (URL) to an *Eclipse* software repository. You can find all releases of *Sirius* [here](#). Choose a version of *Sirius* and follow the link to the update site.

**Step 2:** Copy the URL of the update site from your browser. Depending on the version of *Sirius* that you would like to install, this URL could look like this:

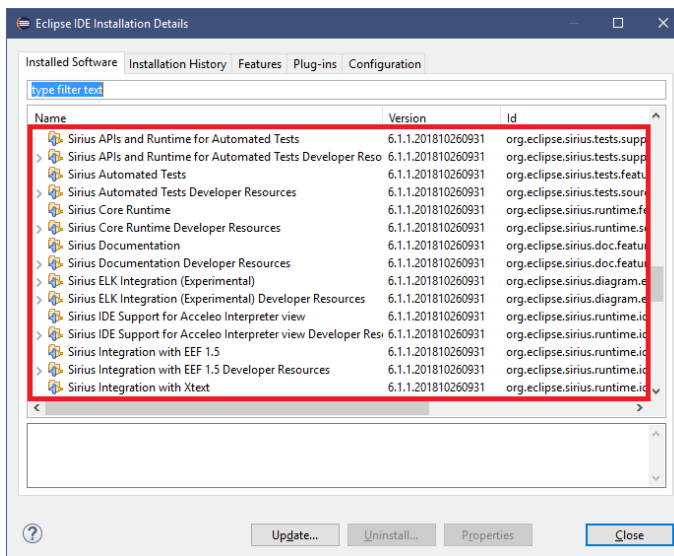
*<https://download.eclipse.org/sirius/updates/releases/6.1.1/photof/>*

Open *Eclipse*. From the menu bar, choose **Help → Install New Software**. Paste the URL of the update site into the “**Work with**”-input-box and click **Add... → Add**.



**Step 3:** Check all four categories to install *Sirius* completely (as shown above). Click **Next**. *Eclipse* will now check requirements and dependencies. After that, click **Next → Finish**. *Eclipse* will now install all software components of *Sirius* in the background. When the installation process has finished, you will have to restart *Eclipse*.

To check if the installation was successful, open *Eclipse* and navigate to **Help → About Eclipse IDE → Installation Details → Installed Software**.



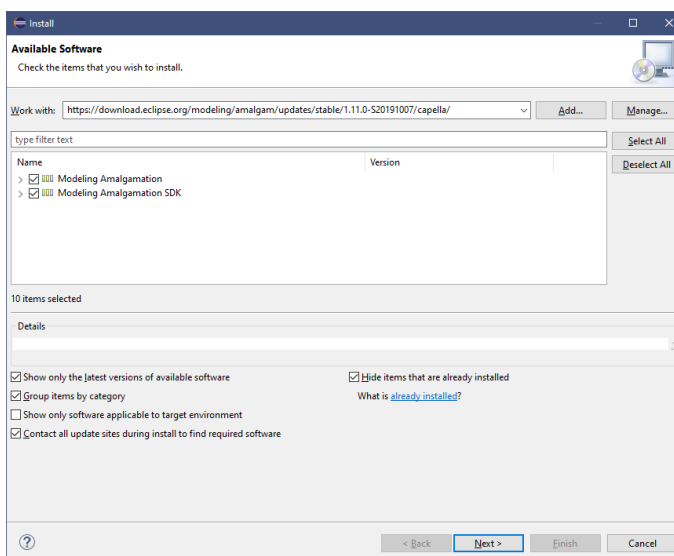
### 2.2.4. Setting up Amalgam

**Step 1:** First you will need a link (URL) to an *Eclipse* software repository. You can find all releases of *Amalgam* [here](#). Choose a version of *Amalgam* and follow the link to the update site.

**Step 2:** Copy the URL of the update site from your browser. Depending on the version of *Amalgam* that you would like to install, this URL could look like this:

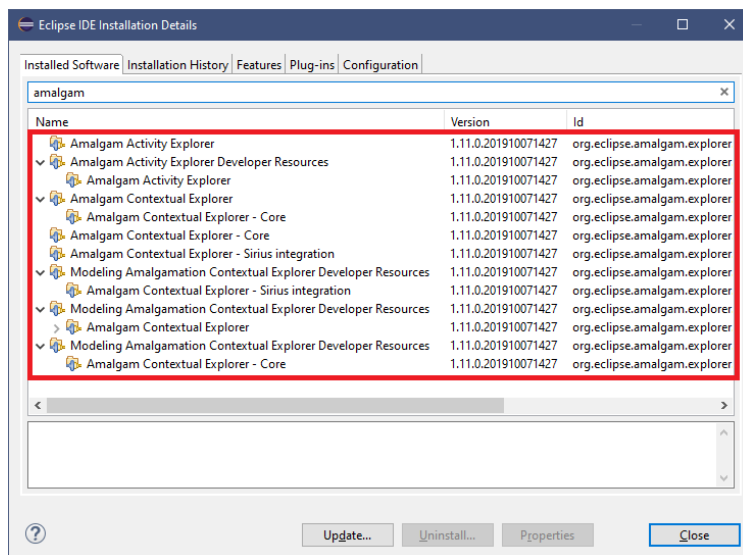
*https://download.eclipse.org/modeling/amalgam/updates/stable/1.11.0-S20191007/capella/*

Open *Eclipse*. From the menu bar, choose **Help** → **Install New Software**. Paste the URL of the update site into the “**Work with**”-input-box and click **Add... → Add**.



**Step 3:** Check the two categories to install *Amalgam* completely (as shown above). Click Next. *Eclipse* will now check requirements and dependencies. After that, click **Next → Finish**. *Eclipse* will now install all software components of *Amalgam* in the background. When the installation process has finished, you will have to restart *Eclipse*.

To check if the installation was successful, open *Eclipse* and navigate to **Help → About Eclipse IDE → Installation Details → Installed Software**.



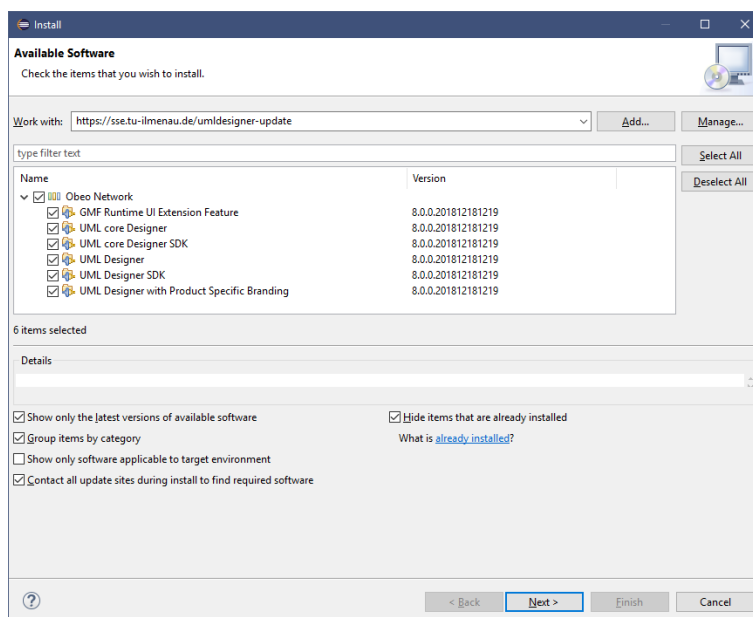
## 2.2.5. Setting up UMLDesigner

*UMLDesigner* is a tool for graphical editing of UML models in *Eclipse Modelling Framework*.

**Step 1:** Copy the following URL of the update site for *UMLDesigner*.

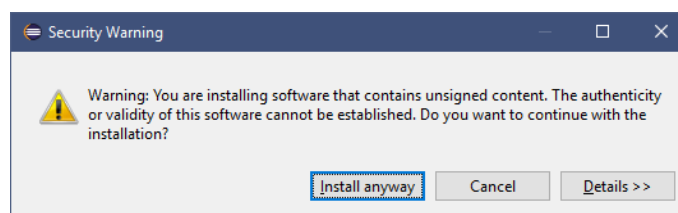
*<https://sse.tu-ilmenau.de/umldesigner-update>*

Open *Eclipse*. From the menu bar, choose **Help** → **Install New Software**. Paste the URL of the update site into the “**Work with**”-input-box and click **Add...** → **Add**.

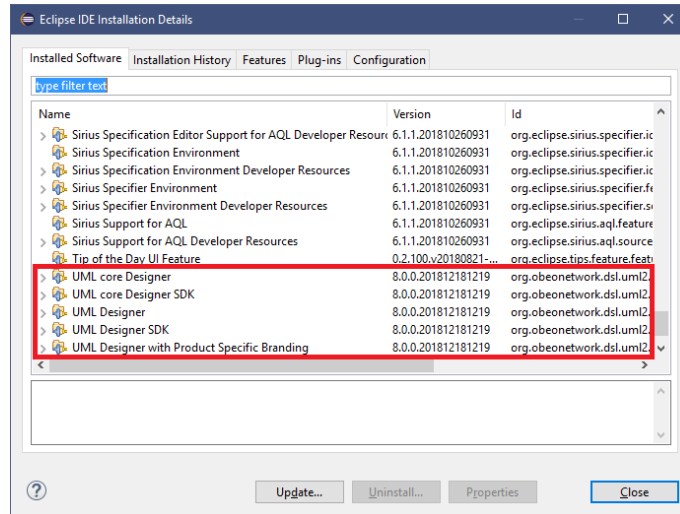


**Step 2:** Check the category **Obeo Network** to install *UMLDesigner* completely (as shown above). Click **Next**. *Eclipse* will now check requirements and dependencies. After that, click **Next** → **Finish**. *Eclipse* will now install all software components of *UMLDesigner* in the background. When the installation process has finished, you will have to restart *Eclipse*.

If *Eclipse* shows a message: “Warning: You are installing software that contains unsigned content”, click **Install anyway**.



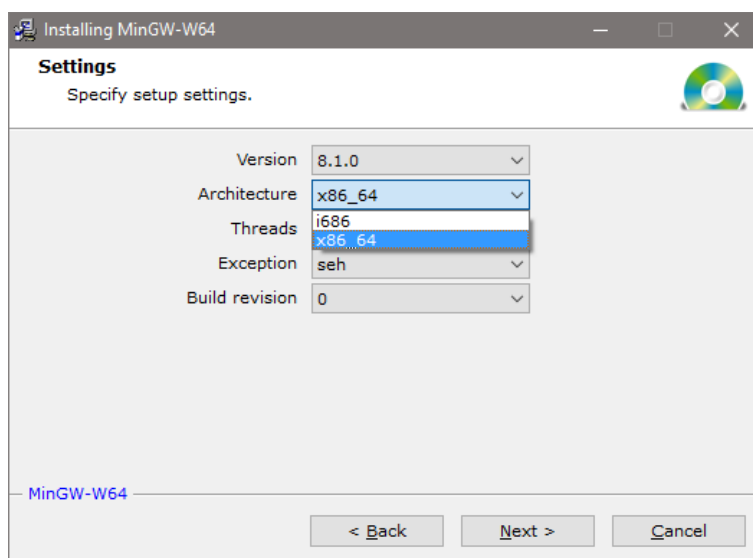
**Step 3:** Check if the installation was successful. Open *Eclipse* and navigate to **Help** → **About Eclipse IDE** → **Installation Details** → **Installed Software**.



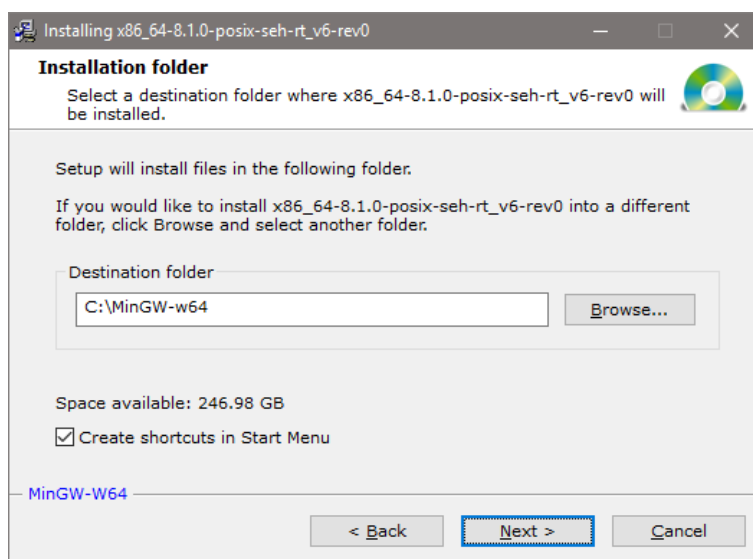
## 2.3. Setting up MinGW-Compiler

**Step 1:** Download an installation file of your choice. For MDE4CPP the 64-bit version of the 'traditional' *MinGW-Compiler* called *MinGW-w64* is required. You can download it [here](#) (online installer).

**Step 2:** Start the installer. After clicking **Next** on the initial dialog, choose your system settings. Choose to install *MinGW-w64* as 64-bit version. We will leave all other settings as is.

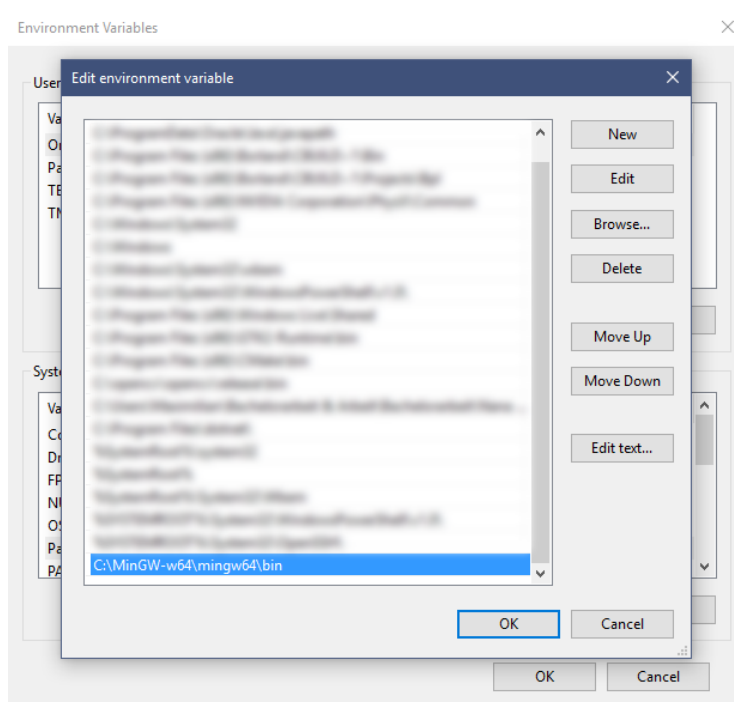


**Step 3:** After clicking **Next** on the settings dialog, choose the installation path. It is highly recommended to choose **C:\MinGW-w64** as the installation directory.



**Step 4:** After setting up the installation directory, the installation process will begin. After the process finishes, click **Next** to finish the setup.

**Step 5:** Add *MinGW-w64* to the environment variables. Open a file explorer and right-click on **This PC**. Click on **Properties** → **Advanced System Settings** → **Environment Variables**. Under **System variables**, find the **Path** variable. Click on **Edit...** → **New** and enter the *MinGW-w64* binaries directory. In our case the directory is **C:\MinGW-w64\mingw64\bin**. Click OK to finish.



**Hint:** You can check if your setup was successful by opening a command prompt and typing in `gcc -v`. You should see an output similar to this:

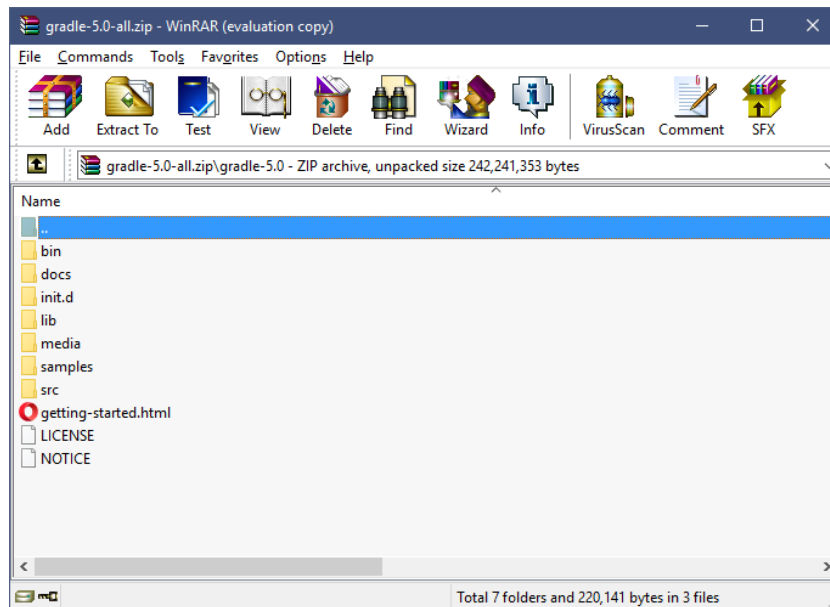
```

C:\> gcc -v
Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=C:/MinGW-w64/mingw64/bin/./libexec/gcc/x86_64-w64-mingw32/8.1.0/lto-wrapper.exe
Target: x86_64-w64-mingw32
Configured with: ../../src/gcc-8.1.0/configure --host=x86_64-w64-mingw32 --build=x86_64-w64-mingw32 --target=x86_64-w64-mingw32 --prefix=/mingw64 --with-sysroot=/c/mingw810/x86_64-810-posix-seh-rt_v6-rev0/mingw64 --enable-shared --enable-static --disable-multilib --enable-languages=c,c++,fortran,lto --enable-libstdcxx-time=yes --enable-threads=posix --enable-libgomp --enable-libatomic --enable-lto --enable-graphite --enable-checking=release --enable-fully-dynamic-string --enable-version-specific-runtime-libs --disable-libstdcxx-pch --disable-libstdcxx-debug --enable-bootstrap --disable-rpath --disable-win32-registry --disable-nls --disable-werror --disable-symvers --with-gnu-as --with-gnu-ld --with-arch=nocona --with-tune=core2 --with-libiconv --with-system-zlib --with-gmp=/c/mingw810/prerequisites/x86_64-w64-mingw32-static --with-mpfr=/c/mingw810/prerequisites/x86_64-w64-mingw32-static --with-mpc=/c/mingw810/prerequisites/x86_64-w64-mingw32-static --with-isl=/c/mingw810/prerequisites/x86_64-w64-mingw32-static --with-pkgversion='x86_64-posix-seh-rev0, Built by MinGW-W64 project' --with-bugurl=https://sourceforge.net/projects/mingw-w64 CFLAGS='-O2 -pipe -fno-ident -I/c/mingw810/x86_64-810-posix-seh-rt_v6-rev0/mingw64/opt/include -I/c/mingw810/prerequisites/x86_64-zlib-static/include -I/c/mingw810/prerequisites/x86_64-w64-mingw32-static/include' CXXFLAGS='-O2 -pipe -fno-ident -I/c/mingw810/x86_64-810-posix-seh-rt_v6-rev0/mingw64/opt/include -I/c/mingw810/prerequisites/x86_64-zlib-static/include -I/c/mingw810/prerequisites/x86_64-w64-mingw32-static/include' CPPFLAGS='-I/c/mingw810/x86_64-810-posix-seh-rt_v6-rev0/mingw64/opt/include -I/c/mingw810/prerequisites/x86_64-zlib-static/include -I/c/mingw810/prerequisites/x86_64-w64-mingw32-static/include' LDFLAGS='-pipe -fno-ident -L/c/mingw810/x86_64-810-posix-seh-rt_v6-rev0/mingw64/opt/lib -L/c/mingw810/prerequisites/x86_64-zlib-static/lib -L/c/mingw810/prerequisites/x86_64-w64-mingw32-static/lib '
Thread model: posix
gcc version 8.1.0 (x86_64-posix-seh-rev0, Built by MinGW-W64 project)
C:\>

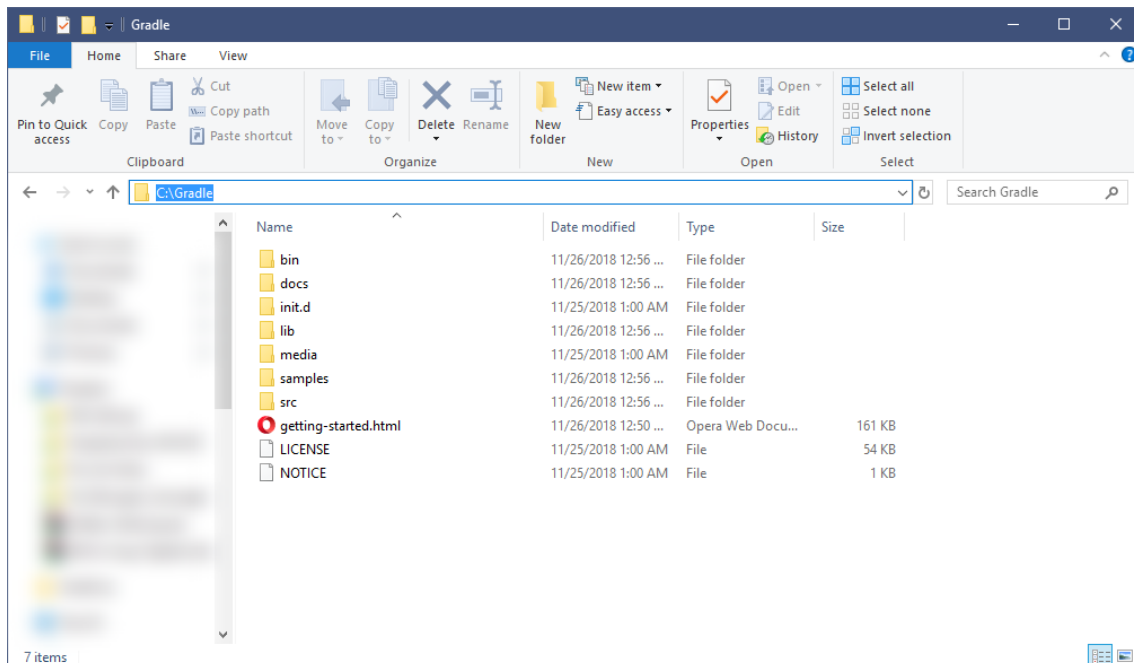
```

## 2.4. Setting up Gradle

**Step 1:** Download *Gradle*. You can download a packaged version [here](#). For this guide, we choose the **Complete** packaged .ZIP archive. After downloading *Gradle*, open the archive (e.g. using *WinRAR*).

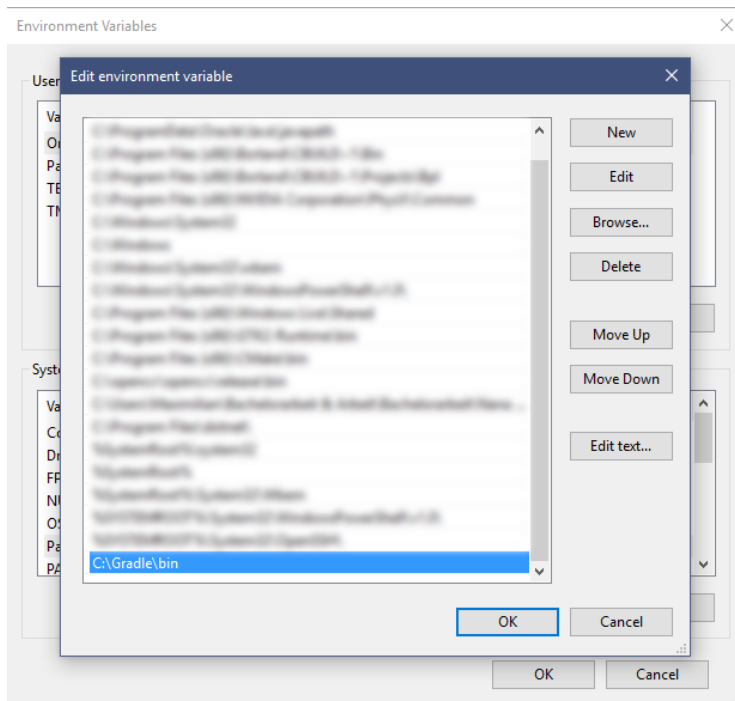


**Step 2:** Choose a directory to extract *Gradle*. It is highly recommended to extract *Gradle* to **C:\Gradle**. Create the directory and extract the files.

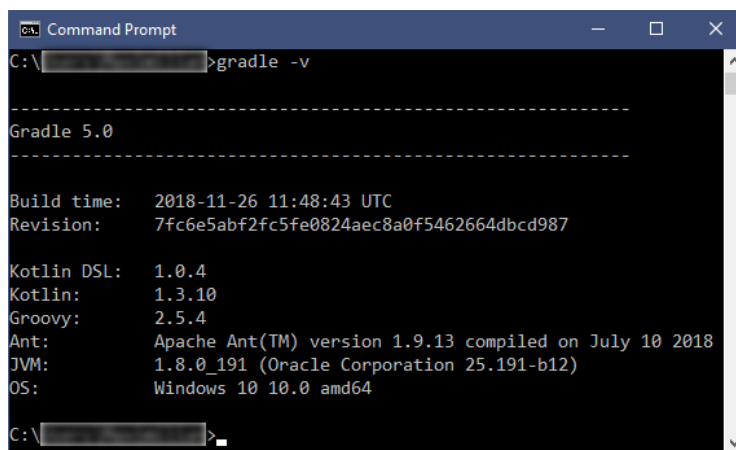




**Step 3:** Add *Gradle* to the environment variables. Open a file explorer and right-click on **This PC**. Click on **Properties** → **Advanced System Settings** → **Environment Variables**. Under **System variables**, find the **Path** variable. Click on **Edit...** → **New** and enter the *Gradle* binaries directory. In our case the directory is **C:\Gradle\bin**. Click OK to finish.

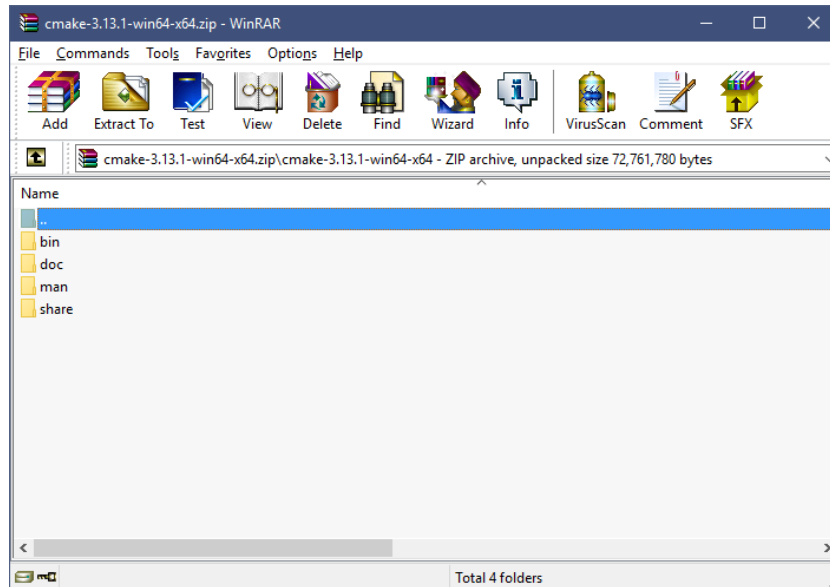


**Hint:** You can check if your setup was successful by opening a command prompt and typing in `gradle -v`. You should see an output similar to this:

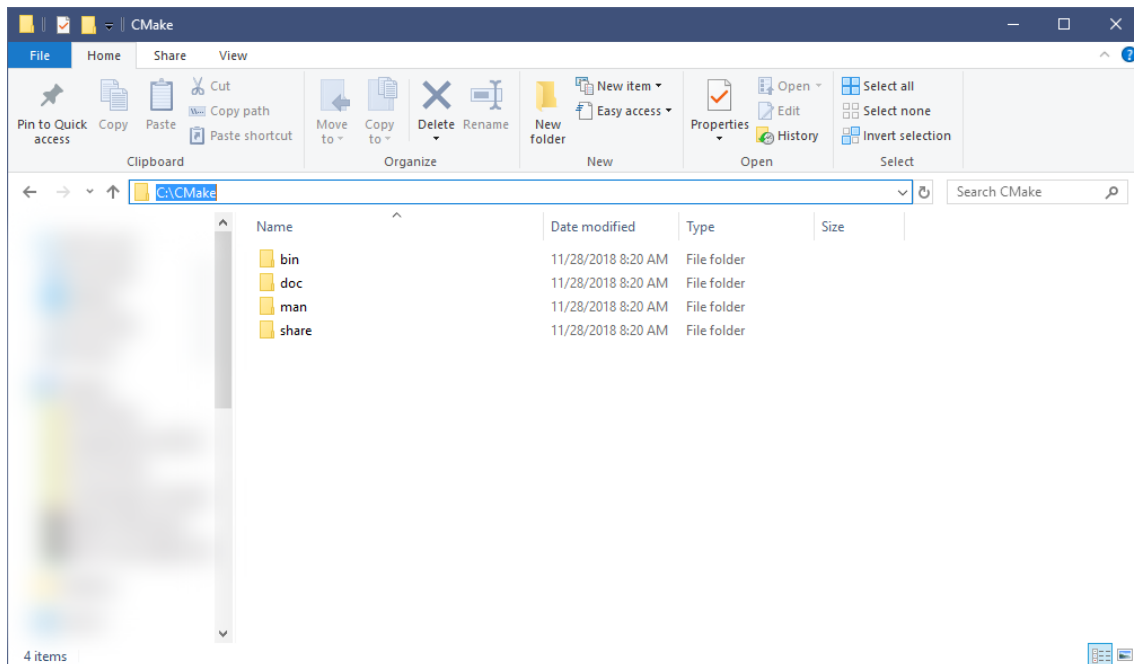


## 2.5. Setting up CMake

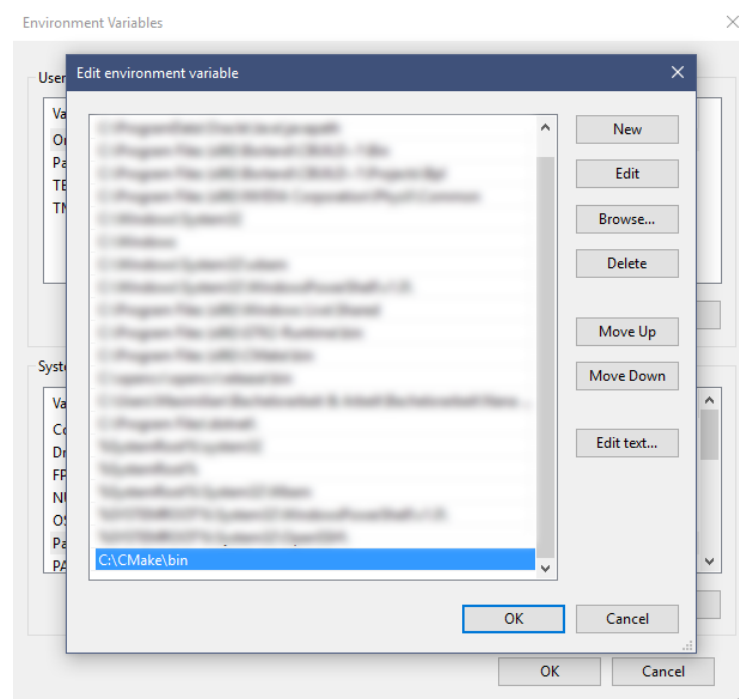
**Step 1:** Download *CMake*. You can download a packaged version or an installer suitable for your system [here](#). **Note:** For this guide, we choose a 64-bit-version .ZIP archive which does not require installation. After downloading *CMake*, open the archive (e.g. using *WinRAR*).



**Step 2:** Choose a directory to extract *CMake*. It is highly recommended to extract *CMake* to **C:\CMake**. Create the directory and extract the files.



**Step 3:** Add *CMake* to the environment variables. Open a file explorer and right-click on **This PC**. Click on **Properties** → **Advanced System Settings** → **Environment Variables**. Under **System variables**, find the **Path** variable. Click on **Edit...** → **New** and enter the *CMake* binaries directory. In our case the directory is **C:\CMake\bin**. Click OK to finish.



**Hint:** You can check if your setup was successful by opening a command prompt and typing in `cmake --version`. You should see an output similar to this:

```
Command Prompt
C:\>cmake --version
cmake version 3.13.1

CMake suite maintained and supported by Kitware (kitware.com/cmake).
C:\>
```

## 3. Getting MDE4CPP

There are two ways of how to obtain MDE4CPP and get the files to your local machine. Either via a packaged .ZIP archive or by directly linking to the repository using the *Git* software.

Although you can download an archive, it is **heavily** recommended (for development purposes and/or to easily update the project files) to work with the repository via *Git*.

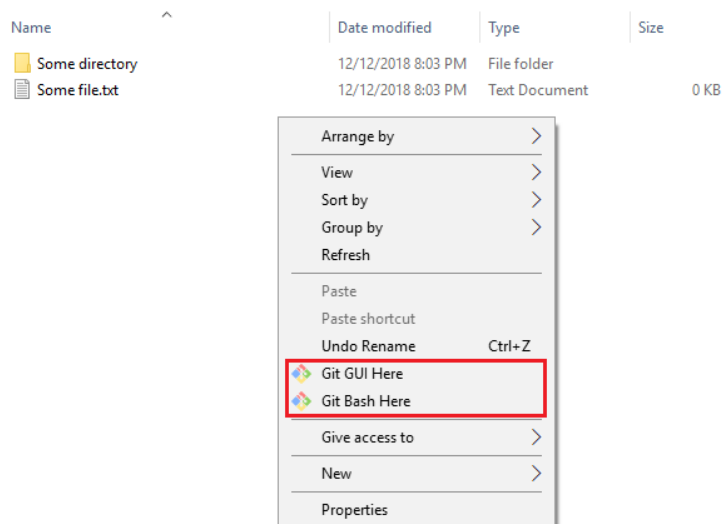
### 3.1. Setting up Git

**Step 1:** Download the installation file for *Git*. Choose an installation file suitable for your system [here](#).

**Step 2:** Start the installer. After clicking **Next** on the initial dialog, choose any installation path for *Git* (e.g. **C:\Program Files\Git**). Using the following few dialogs, choose your installation settings. The *Git* installer offers several options for customized installation. For this guide, we leave everything as is (for every dialog).

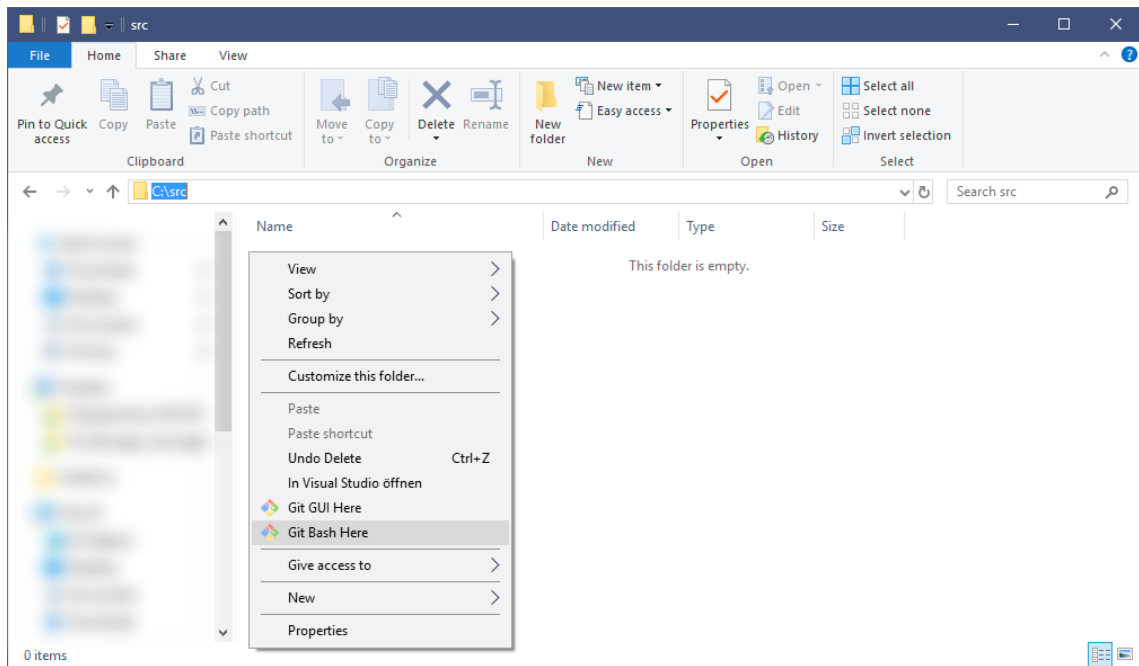
After clicking **Install** on the last installation settings dialog, wait for the installation process to finish.

**Hint:** You can check if your installation was successful by opening a file explorer and navigating to any directory of your choice. Navigate the cursor to a blank spot and right-click. The context menu should look like this:



## 3.2. Cloning the Repository

**Step 1:** After having *Git* installed, choose a directory that you want to clone the MDE4CPP repository to. For this guide, we choose **C:\src**. Navigate to your MDE4CPP directory and right-click. Choose **Git Bash Here**



**Step 2:** The **Git Bash** should have opened. Type in the following command:

```
git clone https://github.com/MDE4CPP/MDE4CPP.git
```

Press Enter. *Git* will now clone the repository into the MDE4CPP directory. In our case the directory is **C:\src\MDE4CPP**. You should get an output similar to this:

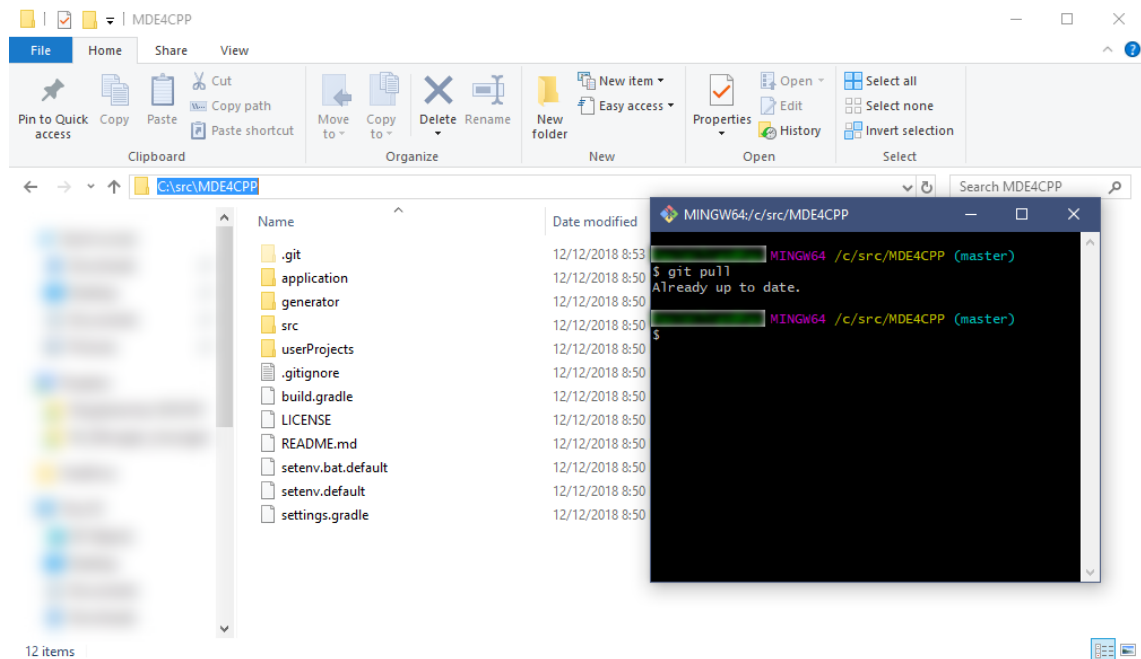
```
MINGW64/c/src
MINGW64 /c/src
$ git clone https://github.com/MDE4CPP/MDE4CPP.git
Cloning into 'MDE4CPP'...
remote: Enumerating objects: 885, done.
remote: Counting objects: 100% (885/885), done.
remote: Compressing objects: 100% (490/490), done.
remote: Total 50528 (delta 642), reused 521 (delta 335), pack-reused 49643
Receiving objects: 100% (50528/50528), 36.74 MiB | 402.00 KiB/s, done.
Resolving deltas: 100% (35358/35358), done.
Checking out files: 100% (1648/1648), done.
MINGW64 /c/src
$ |
```

**Step 3:** You have now successfully cloned the MDE4CPP repository on your local machine. To pull the latest state of the repository later, navigate to your MDE4CPP directory (in our case the directory is **C:\src\MDE4CPP**) and open a **Git Bash** (see above). Type in the following command:

```
git pull
```

*Git* will now either tell you if your repository state is up to date or download the latest changes.

If you want to use MDE4CPP for development purposes, please find further information about *Git* on the [documentation page](#).



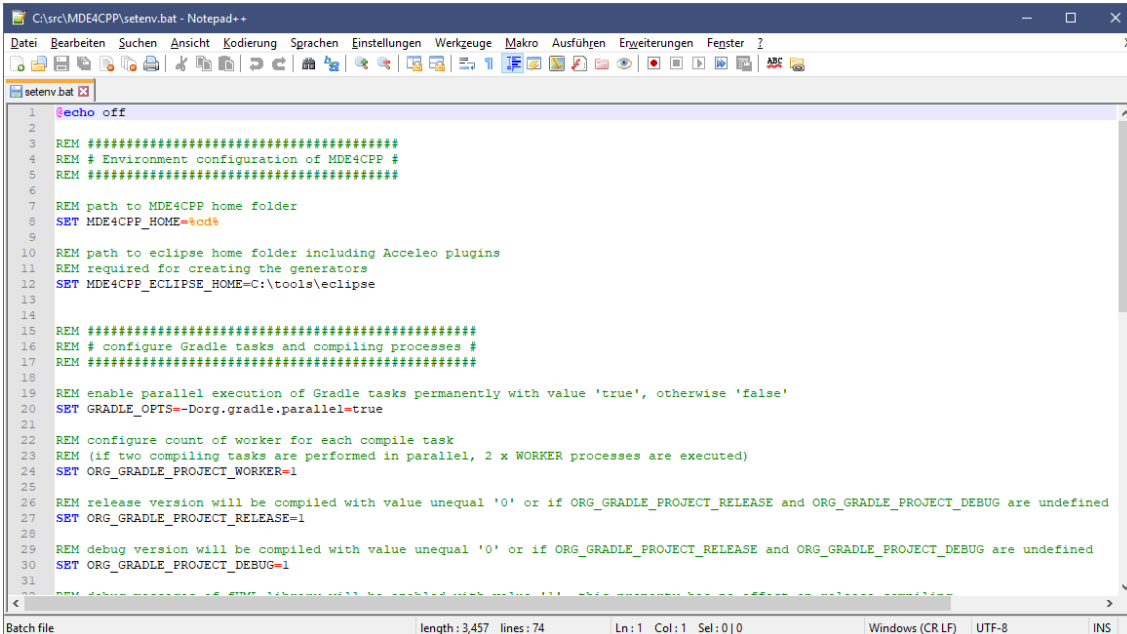
## 4. Setting up MDE4CPP

### 4.1. Environment Configuration

Before being able to build the project, you first have to configure some environment parameters which are required for the build process.

**Step 1:** Navigate to your MDE4CPP directory (in our case the directory is **C:\src\MDE4CPP**).

Find a file called **setenv.bat.default**, select it and press **CTRL+C** and **CTRL+V** to create a backup of the original file (in case something goes wrong). Rename the original file from **setenv.bat.default** to **setenv.bat** and open it with a text editor of your choice (e.g. [Notepad++](#)).



```

1  @echo off
2
3  REM #####
4  REM # Environment configuration of MDE4CPP #
5  REM #####
6
7  REM path to MDE4CPP home folder
8  SET MDE4CPP_HOME=%cd%
9
10 REM path to eclipse home folder including Acceleo plugins
11 REM required for creating the generators
12 SET MDE4CPP_ECLIPSE_HOME=C:\tools\eclipse
13
14
15 REM #####
16 REM # configure Gradle tasks and compiling processes #
17 REM #####
18
19 REM enable parallel execution of Gradle tasks permanently with value 'true', otherwise 'false'
20 SET GRADLE_OPTS=-Dorg.gradle.parallel=true
21
22 REM configure count of worker for each compile task
23 REM (if two compiling tasks are performed in parallel, 2 x WORKER processes are executed)
24 SET ORG_GRADLE_PROJECT_WORKER=1
25
26 REM release version will be compiled with value unequal '0' or if ORG_GRADLE_PROJECT_RELEASE and ORG_GRADLE_PROJECT_DEBUG are undefined
27 SET ORG_GRADLE_PROJECT_RELEASE=1
28
29 REM debug version will be compiled with value unequal '0' or if ORG_GRADLE_PROJECT_RELEASE and ORG_GRADLE_PROJECT_DEBUG are undefined
30 SET ORG_GRADLE_PROJECT_DEBUG=1
31
32 REM debug messages of GINX library will be enabled with value 1. This property has no effect on release builds

```

**Step 2:** Now you will need to replace some lines inside of the file. The lines that need to be adapted will be listed down below. The replaced directories or version numbers will be the ones chosen in the installation guides in section 2. Please replace with your OWN parameters, if deviant.

replace `SET MDE4CPP_HOME=%cd%`

with `SET MDE4CPP_HOME=C:\src\MDE4CPP`

replace `SET MDE4CPP_ECLIPSE_HOME=C:\tools\eclipse`

with `SET MDE4CPP_ECLIPSE_HOME=C:\eclipse\Modelling`

replace *SET CMAKE\_HOME=C:\tools\CMake*

with *SET CMAKE\_HOME=C:\CMake*

replace *SET COMPILER\_HOME=C:\tools\MinGW*

with *SET COMPILER\_HOME=C:\MinGW-w64\mingw64*

replace *SET JAVA\_HOME=C:\Program Files\java\jdk1.8.0\_162*

with *SET JAVA\_HOME=C:\Program Files\Java\jdk1.8.0\_191*

replace *REM SET PATH=C:\Windows\system32;  
C:\Windows;C:\Windows\System32\Wbem;  
C:\Windows\System32\WindowsPowerShell\v1.0|*

with *SET PATH=C:\Windows\system32;  
C:\Windows;C:\Windows\System32\Wbem;  
C:\Windows\System32\WindowsPowerShell\v1.0|*

replace *SET COMPILER\_VERSION=8.1.0*

**Hint:** You can find out the version of your compiler by opening a command prompt and typing in: *gcc -v* (see section 2.3).

**Step 3:** Make sure to save the file ***setenv.bat*** after you applied the changes.



## 4.2. Building the Project

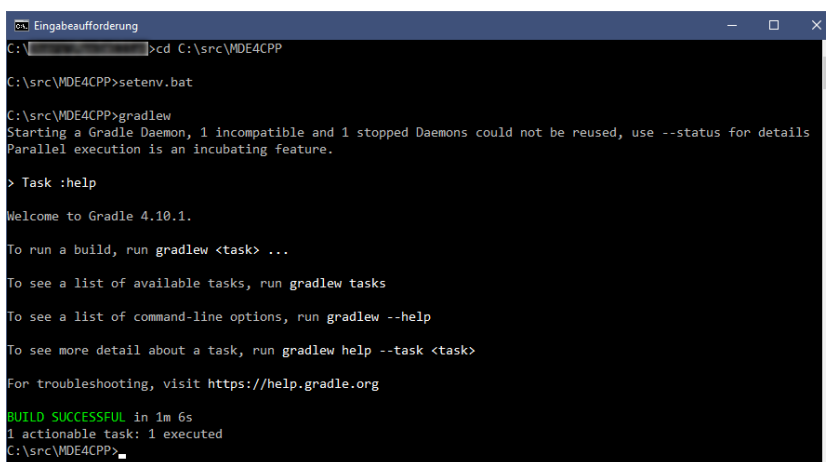
**Step 1:** Open a command prompt, change the directory to your MDE4CPP directory (in our case the directory is **C:\src\MDE4CPP**) and run the adapted **setenv.bat**. Type in the following commands:

```
cd C:\src\MDE4CPP
```

```
setenv.bat
```

```
gradlew
```

You should see an output similar to this:



```
cmd - Eingabeaufforderung
C:\>cd C:\src\MDE4CPP
C:\src\MDE4CPP>setenv.bat
C:\src\MDE4CPP>gradlew
Starting a Gradle Daemon, 1 incompatible and 1 stopped Daemons could not be reused, use --status for details
Parallel execution is an incubating feature.

> Task :help

Welcome to Gradle 4.10.1.

To run a build, run gradlew <task> ...

To see a list of available tasks, run gradlew tasks

To see a list of command-line options, run gradlew --help

To see more detail about a task, run gradlew help --task <task>

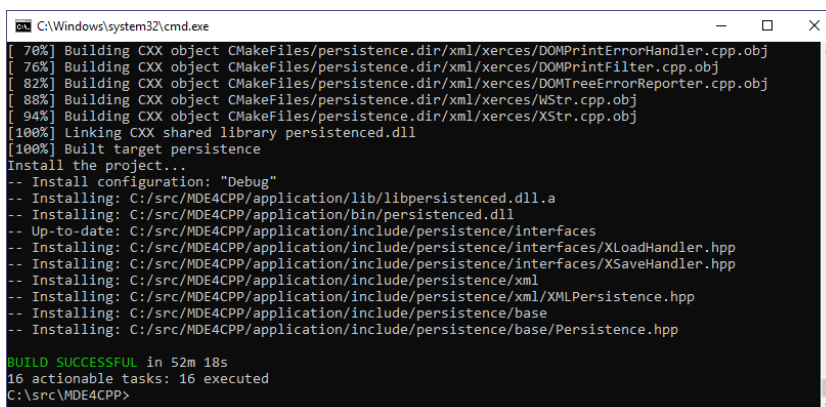
For troubleshooting, visit https://help.gradle.org

BUILD SUCCESSFUL in 1m 6s
1 actionable task: 1 executed
C:\src\MDE4CPP>
```

**Step 2:** In the same the command prompt, type in the following command to start the building process (this might take a while):

```
gradlew buildAll
```

After the compilation process has finished, you should see an output similar to this:



```
cmd - C:\Windows\system32\cmd.exe
[ 70%] Building CXX object CMakeFiles/persistence.dir/xml/xerces/DOMPrintErrorHandler.cpp.obj
[ 76%] Building CXX object CMakeFiles/persistence.dir/xml/xerces/DOMPrintFilter.cpp.obj
[ 82%] Building CXX object CMakeFiles/persistence.dir/xml/xerces/DOMTreeErrorReporter.cpp.obj
[ 88%] Building CXX object CMakeFiles/persistence.dir/xml/xerces/WStr.cpp.obj
[ 94%] Building CXX object CMakeFiles/persistence.dir/xml/xerces/XStr.cpp.obj
[100%] Linking CXX shared library persistence.dll
[100%] Built target persistence
Install the project...
-- Install configuration: "Debug"
-- Installing: C:/src/MDE4CPP/application/lib/libpersistence.dll.a
-- Installing: C:/src/MDE4CPP/application/bin/persistence.dll
-- Up-to-date: C:/src/MDE4CPP/application/include/persistence/interfaces
-- Installing: C:/src/MDE4CPP/application/include/persistence/interfaces/XLoadHandler.hpp
-- Installing: C:/src/MDE4CPP/application/include/persistence/interfaces/XSaveHandler.hpp
-- Installing: C:/src/MDE4CPP/application/include/persistence/xml
-- Installing: C:/src/MDE4CPP/application/include/persistence/xml/XMLPersistence.hpp
-- Installing: C:/src/MDE4CPP/application/include/persistence/base
-- Installing: C:/src/MDE4CPP/application/include/persistence/base/Persistence.hpp

BUILD SUCCESSFUL in 52m 18s
16 actionable tasks: 16 executed
C:\src\MDE4CPP>
```

**Finished:** MDE4CPP was successfully set up and is now ready for use.