Stageverslag

Python Framework Installer

Pieter-Jan Robrecht

Student Industrieel ingenieur ICT
01/08/2016 - 19/08/2016

Inhoudsopgave

Listings

1 Inleiding

In het kader van mijn masterproef was het mogelijk om bij Televic Rail een stage te lopen tijdens de zomer. Gedurende deze stage heb ik mijn masterproef voorbereid en onderzoek gedaan naar de verschillende mogelijkheden die er zijn voor het uitvoeren van mijn opdracht. De opdracht die voltooid moet worden is de volgende:

> Televic Rail heeft een Python test framework ontworpen. Dit framework werd oorspronkelijk gebruikt op een volledig uitgeruste testtoren, maar het framework werd aangepast zodanig dat het onafhankelijk van de testtoren gebruikt kan worden. Aangezien het framework gebruik maakt van verschillende niet-standaard Python bibliotheken, is het installeren van het framework op een nieuw systeem een hele klus. Het doel is dan ook het maken van een installer-updater waarmee dit proces kan worden vergemakkelijkt zodanig dat er zo min mogelijk interactie van de gebruiker moet zijn.

Tijdens de duur van de stage werden de verschillende implementatie manieren onderzocht en werd er een algemene architectuur voor de installer-updater uitgedacht. In wat volgt wordt er beschreven welke stappen er werden ondernomen om het probleem onder te verdelen in enkele logische onderdelen en hoe deze het probleem dan oplossen.

2 Analyse van het probleem

Tijdens het bespreken van het probleem werd het al snel duidelijk dat er verschillende scenario's zijn waarin het framework gebruikt wordt. Het framework wordt als een standalone programma gebruikt op een laptop of desktop maar het zal ook moeten draaien op verschillende testtorens. Voor de verschillende omgevingen zal iedere keer een andere configuratie nodig zijn maar beiden omgevingen gebruiken Windows als besturingssysteem. Uiteraard moeten we ervan uitgaan dat deze situatie slechts tijdelijk is. Er wordt best een applicatie geschreven die op meer dan enkel Windows kan draaien.

Tijdens het installeren van het framework moeten we rekening houden met het feit dat iedere computer een andere configuratie en andere drivers zal nodig hebben. We kunnen de gebruiker vragen om de verschillende drivers te selecteren, maar, mocht het mogelijk zijn, dan zou de applicatie best de verschillende devices detecteren om vervolgens zelf de juiste drivers aan te vinken in de volledige lijst van drivers. Zo moet de gebruiker minder weet hebben van welke drivers er allemaal nodig zijn. Het probleem van de drivers beperkt zich jammer genoeg niet enkel tot dit. Als er nieuwe hardware ontwikkelt wordt, zullen er ook verschillende nieuwe drivers nodig zijn. De lijst van beschikbare drivers zal dan tijdig moeten worden aangepast. Er zal dus ergens een volledige lijst moeten zijn met alle devices en welke drivers eraan gekoppeld zijn. Deze lijst zou dan eventueel ook de laatste versienummers hebben. De updater kan deze lijst dan ook gebruiken om te controleren of de versie van de geïnstalleerde drivers de laatste versie is.

De installer is best ook een programma dat zo eenvoudig mogelijk uit te breiden of aan te passen is zodanig dat het in de toekomst nog kan worden aangepast. De interface naar de gebruiker toe is best ook zo eenvoudig mogelijk zodanig dat er geen problemen kunnen ontstaan tijdens de installatie/update van het framework. Uiteraard gaan we er vanuit moeten gaan dat er af en toe

een probleem zal ontstaan tijdens de installatie. De installatie/updates zullen dus dan in een afgesloten omgeving moeten gebeuren en na het correct uitvoeren in het grote geheel geplaatst worden.

De updater bezit gelijkaardige problemen. We zullen een manier moeten zoeken waarmee we gemakkelijk kunnen controleren naar de versienummers van de software. De gebruiker zal ook de optie moeten krijgen om de update uit te voeren. De gebruiker moet zelf beslissen wanneer de update uitgevoerd zal worden zodanig dat de update gebeurd als de gebruiker er klaar voor is.

3 Oplossingen

Nu we een algemeen beeld hebben van wat we juist allemaal moeten voorzien, kunnen we vervolgens aan de slag met het zoeken naar een goede oplossing voor alle problemen. In wat volgt gaan we alle mogelijkheden overlopen die gebruikt kunnen worden om de installer-updater te implementeren. Alle verschillende opties gaan worden overlopen en de verschillende voor- en nadelen zullen besproken worden. Niet alle oplossingen die gaan worden aangeboden bevatten een updater en installer oplossing. Daarom zullen we van hieruit een opsplitsing maken tussen de installer en de updater. Voor beiden zijn er oplossingen aanwezig en er zal gekeken moeten worden welke combinaties mogelijk zijn.

3.1 Mogelijkheden

Laten we eerst kijken naar de besturingssysteem-onafhankelijkheid van het framework. De besturingssysteem-onafhankelijkheid van Pyhton hangt af van de code en van de bibliotheken die worden gebruikt. Zolang deze bibliotheken onafhankelijk zijn van het besturingssysteem is er geen enkel probleem. Java biedt ook een oplossing aan voor dit probleem aangezien Java ook volledig besturingssysteem-onafhankelijk is. Jammer genoeg zijn er amper tot geen programma's die een installer/updater kunnen genereren in Java. Alle code voor zo'n programma zal zelf geschreven moeten worden.

3.1.1 Qt Installer Framework [8]

Het Qt framework zou in staat zijn om installers te maken die op verschillende besturingssystemen zou kunnen draaien [1]. De software zou de gebruiker door het installatie, update en verwijderproces leiden. De programmeur moet enkel de nodige informatie over de te installeren software meegeven. Bij iedere component van de installer kan een script toegevoegd worden zodanig dat de installatie gepersonaliseerd kan worden [9].

Het Framework biedt twee types installers aan: een online en een offline installer. Beide installers zullen een maintenance tool installer die gebruikt kan worden om componenten te updaten, toe te voegen en te verwijderen. Het verschil tussen de twee installers is het volgende: de offline installer zal alle nodige

[1] Linux, Microsoft Windows en OS X [10]

componenten bevatten in de installer zelf terwijl de online versie deze zal downloaden van een repository. Bij de offline installer zullen ook al deze componenten geïnstalleerd worden terwijl bij de online installer de keuze gemaakt kan worden tussen de verschillende componenten [10].

Dit framework lijkt op het eerste zicht als een goede kandidaat maar het installer van het Python testframework is momenteel gepersonaliseerd voor Windows. De installatie van het testframework omvat het installeren van enkele executables die speciaal voor Windows zijn ontworpen. De grote vraag is dan: is het mogelijk om deze executables in een installer te steken die ontworpen is voor Linux, en zal deze installer dan wel werken? De installer zal ook de verschillende zip files moeten kunnen uitpakken en de setup uitvoeren.

3.1.2 WiX [14]

Windows Installer XML Toolset is een toolset waarmee Windows Installer packages gemaakt kunnen worden vanuit XML code. Met de toolset is het mogelijk om een installer te maken met als extensie .msi. De installer zal dus enkel bruikbaar zijn in Windows. Met WiX is het mogelijk om een installer te schrijven die code van derde partijen gebruikt. Het is dus mogelijk om executables te includeren in de installer [15].

Dankzij het patch systeem moet het mogelijk zijn om updates uit te voeren. Er zal wel onderzocht moeten worden of een automatische version check kan uitgevoerd worden bij het opstarten. Het schrijven van een patch voor ieder update die uitkomt is geen goede oplossing. Als er dus geen automatische version check en installatie aanwezig is, dan zal gekeken moeten worden om een updater te includeren in de installer. Deze updater zal dan bij het opstarten van de software zoeken naar de nieuwe updates en eventueel de mogelijkheid geven om te updaten.

3.1.3 NSIS [7]

Nullsoft Scriptable Install System is een opensource systeem waarmee Windows installers kunnen gemaakt worden. Een installer van NSIS zal dus ook enkel kunnen draaien op Windows. NSIS is script-based waardoor verschillende mogelijkheden ingebouwd kunnen worden in de installer. Een installer van NSIS kan ook uitgebreid worden met plug-ins waardoor de functionaliteit kan worden uitgebreid. Deze plug-ins kunnen geschreven worden in verschillende talen zoals C, C++, ... [6]. Het is mogelijk om in de installer verschillende componenten te includeren. Dankzij dit systeem kunnen we executables die al bestaan, includeren in de installer. Of de installer de mogelijkheid bezit om een zip uit te pakken en de setup te doorlopen, zal nog onderzocht moeten worden.

Er moet nog onderzocht worden of NSIS de mogelijkheid aanbiedt om een updater te includeren in de gemaakte installer. Mocht dit niet het geval zijn, dan zal ook bij deze optie gezocht moeten worden om een updater te implementeren.

Naast NSIS is er ook nog Inno Setup. Dit is een zeer gelijkaardige tool en bezit gelijkaardige features [5].

3.1.4 Chocolatey [2]

Chocolatey is een Powershell execution engine dat gebruikt maakt van de NuGet packaging infrastructuur. Het systeem is te vergelijken met de package manager apt-get van Linux. Chocolatey zal alle executables, zips, ... die nodig zijn voor een stuk software encapsuleren zodanig dat alles in één package zit. Of de software de zips dan ook kan uitpakken en de setup starten is niet geweten uit de documentatie [1]. Chocolatey biedt mogelijkheden aan zodanig dat installatie, upgrades en verwijdering van het programma in de handen van de programmeur ligt. Er zal wel nog onderzocht moeten worden hoe het installatie en update systeem werkt. De nodige programma's moeten op de juiste locatie geplaatst worden om een goede werking te verzekeren. Het grootste nadeel van Chocolatey is het feit dat de gebruiker de Powershell moet gebruiken om de componenten up te daten en te installeren. Aangezien alles in de Powershell gebeurd moet het wel mogelijk zijn dat er batch files gemaakt worden die deze taken uitvoeren. Deze zouden dan een deel van het werk van de gebruiker kunnen wegnemen. Net zoals de vorige oplossingen is deze ook enkel toepasbaar voor Windows.

3.1.5 WinSparkle [12]

Aangezien voor sommige mogelijkheden, zoals NSIS en WiX, niet geweten is of het update systeem voldoende is, zal er gezocht moeten worden naar alternatieven om een updater te includeren in de installer. Een mogelijke oplossing is WinSparkle. WinSparkle is enkel bruikbaar voor Windows dus deze zou, als dit mogelijk is, in combinatie met NSIS of WiX gebruikt kunnen worden. Jammer genoeg biedt de documentatie van WinSparkle niet veel antwoorden aan [11]. Doordat er niet veel documentatie aanwezig is, kan de implementatie van WinSparkle in de installer lastig zijn. In de documentatie wordt ook het volgende vermeld [13]:

> Update downloads the new version's installer and launches it.

Het is dan ook niet duidelijk of er voor ieder update dan een nieuwe installer moet geschreven worden. Mocht dit het geval zijn, dan is dit een minder aangewezen oplossing aangezien dit voor veel overbodig werk zorgt.

3.1.6 esky [3]

Nog een optie om een updater te implementeren is esky. Esky is een auto-update framework voor frozen Python applicaties. Dankzij een eenvoudige API kunnen applicaties updates zoeken, ophalen en installeren. Er zit ook een mechanisme in dat de applicatie gaat beschermen tegen failed updates. Het is zeker mogelijk om de code van een programma up te daten met dit framework. Wat niet duidelijk is, is in hoeverre het mogelijk is om verschillende componenten up-te-daten. De updater zal ook de Python code omvormen tot een executable. Deze zijn jammer genoeg enkel bruikbaar op Windows.

3.1.7 Google Omaha [4]

Eén van de laatste opties om een updater te maken is Google Omaha. Het is de open-source versie van Google Update. Er is jammer genoeg niet veel te vinden over de mogelijkheden van Google Omaha en er zal dus uitgezocht moeten worden welke mogelijkheden deze aanbied.

Alle gevonden opties zijn op dit punt algemeen uitgelegd. De documentatie van sommige opties was jammer genoeg niet altijd voldoende om een beeld te krijgen wat de optie precies allemaal kan. Daarom gaan we van de meesten opties een HelloWorld applicatie maken. Met deze applicatie kunnen we dan achterhalen hoe eenvoudig het is om deze tool te gebruiken maar wat de mogelijkheden van deze tool zijn. We gaan tijdens het ontwikkelen van de HelloWorld applicaties ook zoeken naar tools waarmee we Windows executables kunnen draaien op Linux.

3.2 Testen

Zoals eerder vermeld gaan we nu enkele HelloWorld applicaties maken met de tools die we hebben besproken in sectie 3.1 op pagina 3. We gaan met deze tools een applicatie maken met als doel een executable in te pakken samen met een zip. De installer moet dan beide componenten correct installeren. Sommige tools bezitten de mogelijkheid om updates door te voeren. Tijdens deze testen zullen we ook kijken of het mogelijk is om één van de twee componenten up te daten met zo min mogelijk extra code. De code die gebruikt werd is terug te vinden in sectie 4 op pagina 8. De verschillende tutorials zullen ook worden bijgehouden zodanig dat het mogelijk is om het bekomen resultaat te repliceren.

3.2.1 Qt Installer Framework

De eerste optie waarmee een installer geschreven is geweest, is de Qt Installer Framework. Met dit framework was het mogelijk om een installer te maken die bestaat uit verschillende componenten, die elk een speciale installatie kunnen krijgen dankzij de bijhorende scripts. Zo was het dus mogelijk om een executable mee te nemen in de installer en deze op te roepen vanuit de installer. De executable die gebruikt werd, was de installer van Python 2.7 . Dit is dus zelf een installer met een GUI en bepaalde instellingen die moeten goed gezet worden. Deze GUI kan gerust onderdrukt worden aangezien alle vooraf ingestelde instellingen al goed staan. Tijdens het eerste gebruik van deze tool, dit duurde ongeveer 1 dag, was het mij nog niet gelukt om dit te onderdrukken maar volgens mij moet dit wel mogelijk zijn. Naast de executable, heb ik ook een component met een zip bestand meegenomen in de installer. Deze zip zou moeten uitgepakt worden op het doelsysteem en vervolgens zou de setup.py uitgevoerd moeten worden. Het uitpakken van de zip is gelukt maar niet op de meest aangewezen manier. Het doorlopen van de setup.py is ook niet gelukt tijdens het maken van de allereerste installer.

Het schrijven van deze eerste installer was zeker niet eenvoudig maar ik had wel de indruk dat het framework gebruikt kan worden om degelijke installers te maken. Eén dag was jammer genoeg onvoldoende om een zeer uitgebreide analyse van de tool te maken. Desondanks het tijdsgebrek heb ik wel de indruk dat deze tool gebruikt kan worden om goede tools te maken. Jammer genoeg is er niet veel voorbeeld code aanwezig en zal het schrijven van grotere/goede installers tijd vragen.

3.2.2 NSIS

Het maken van een installer met NSIS verliep een pak eenvoudiger in vergelijking met het maken van een installer in Qt. De installer zelf omvat dan ook veel minder de Qt isntaller. Dit komt doordat Qt een volledige maintenance tool installeerd waarmee de installer/componenten kunnen vervangen, geupdated en verwijderd worden. Dit zit allemaal standaard in de installer van Qt. Bij NSIS moet zelf een uninstaller geschreven worden en moet de programmeur dan zelf kijken wat er verwijderd moet worden. De programmeur heeft dus meer werk met het schrijven van een goede installer met uninstall mogelijkheden, maar in ruil hiervoor is het wel eenvoudiger om een installer in elkaar te steken. Deze tool heeft geen mogelijkheden om een updater te implementeren. Er zal dus onderzocht moeten worden of het mogelijk is om de NSIS installer te combineren met een updater.

3.2.3 WiX Toolset

WiX is de derde tool die uitgetest is geweest en direct ook de lastigste om mee te werken tot op heden. Het was relatief eenvoudig om online verschillende stukken code te vinden die nodig waren voor enkele functionaliteiten te implementeren. Jammer genoeg waren dit vaak losstaande stukken code en was het zeer moeilijk om deze stukken in het grotere geheel te plaatsen. WiX zal een installer produceren met als extensie .msi, dit zorgt voor het volgende probleem: het is niet mogelijk om tijdens het uitvoeren van de installer, een tweede installer op te roepen met een .msi extensie. Hierdoor kon dus de installer van Python (een toch wel belangrijke component voor het testframework) niet uitgevoerd worden op hetzelfde moment als de installer van WiX. Het uitpakken van een zip bestand is ook niet gelukt met deze tool. Waarschijnlijk is het wel mogelijk om degelijke installers te maken met deze tool, maar het gebruik van enkel xml om de installer te definiëren maakt het niet eenvoudig. Volgens mij is het dan ook eenvoudiger om tools zoals NSIS te gebruiken. Er is minder tijd nodig om een installer te maken die iets eenvoudig kan en het is gemakkelijker te schrijven.

3.2.4 Chocolatey

Als één van de laatste installer tools is het aan de beurt van Chocolatey om onderzocht te worden. Op het eerste zicht is het gebruik van Chocolatey gelijkaardig aan het gebruik van Qt. De opbouw van de package in Chocolatey is zeer

gelijkaardig aan dat van Qt. Beiden hebben een algemeen bestand waarin de configuratie van de component zit en ze hebben een script dat kan uitgevoerd worden bij het installeren, verwijderen, ... Chocolatey heeft één groot voordeel en dat is de mogelijkheid om de powershell van Windows te gebruiken bij het upgraden, installeren en verwijderen. Hierdoor kan de installatie van iedere component gepersonaliseerd worden naar de noden van deze component. Hiernaast voelt Chocolatey ook aan als een zeer degelijk tool. Voor het verzenden van een update is er een NuGet server nodig waarin de nieuwe versie van de component zit. Deze kan dan gedownload worden bij het upgraden en kan dan ook geïnstallerd worden mocht dit nodig zijn. Het grootste voordeel van Chocolatey is ook zijn grootste nadeel, alle handelingen verlopen via de Command Prompt van Windows. Gebruikers hebben dus geen GUI waarmee ze kunnen interageren, maar het moet wel mogelijk zijn om een GUI rond de Command Prompt te maken met bijvoorbeeld Java.

Nu dat we alle verschillende tools hebben besproken waarmee het mogelijk is om een installer te schrijven, is het nu aan de beurt van de tools om een updater te implementeren. Aangezien sommige installer tools, zoals NSIS, niet de mogelijkheid aanbied om een updater te implementeren, moet er dus gekeken worden naar extra software om updates te realiseren. Tijdens het testen zal er gebruik gemaakt worden van de HelloWorld applicatie die gemaakt geweest is met NSIS. Deze zullen we uitbreiden en we gaan proberen één van de componenten in de installer te vervangen met een nieuwe versie. Gedurende het testen gaan we vooral letten op hoe eenvoudig het is om de update uit te voeren.

3.2.5 Google Omaha

Eén van de eerste tools die we gaan uitproberen om updates uit te proberen is Google Omaha. We beginnen met deze want de opzet tijd voor deze tool zou best wel uitgebreid zijn.

3.3 Architectuur

Nu er geweten wat de verschillende opties aanbieden, kunnen we een beter inschatting maken van hoe de uiteindelijke applicatie eruit moet zijn.

4 Code

Listing 1: Qt config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<Installer>
    <Name>Eerste Test</Name>
    <Version>1.0.0</Version>
    <Title>Playground</Title>
```

```
    <Publisher>PJ Industries</Publisher>
    <!-- Directory name is used in component.xml -->
    <StartMenuDir>PythonFrameWork</StartMenuDir>
    <TargetDir>@HomeDir@/IfwExamples/eigenTest</TargetDir
        ↪ >
    <RemoteRepositories>
        <Repository>
            <Url>http://localhost/repository</Url>
        </Repository>
    </RemoteRepositories>
</Installer>
```

Listing 2: Qt package.xml voor een executable

```
<?xml version="1.0" encoding="UTF-8"?>
<Package>
    <DisplayName>Executable</DisplayName>
    <Description>Python die moet worden geinstaleerd</
        ↪ Description>
    <Version>1.0.2-1</Version>
        <Script>installscript.qs</Script>
        <SortingPriority>100</SortingPriority>
    <ReleaseDate>2015-01-01</ReleaseDate>
    <Default>true</Default>
</Package>
```

Listing 3: Qt installscript.qs voor een executable

```
function Component()
{
    //Default component
        //Script van exe
}

Component.prototype.createOperations = function()
{
    // call default implementation to actually install
        ↪ README.txt!
    component.createOperations();

    if (systemInfo.productType === "windows") {
                component.addOperation("Execute"
                , "msiexec"
                , "/i"
                , "@TargetDir@\\python-2.7.3.msi"
                , "/quiet"
```

```
                    , "UNDOEXECUTE"
                    , "msiexec"
                    , "/qb"
                    , "/x"
                    , "@TargetDir@\\python-2.7.3.msi")
    }
}
```

Listing 4: Qt package.xml voor een zip

```
<?xml version="1.0" encoding="UTF-8"?>
<Package>
    <DisplayName>Zip</DisplayName>
    <Description>Zip die moet worden uitgepakt en
        ↪ uitgevoerd.</Description>
    <Version>1.0.0-1</Version>
        <Script>installscript.qs</Script>
        <SortingPriority>90</SortingPriority>
    <ReleaseDate>2015-01-01</ReleaseDate>
    <Default>true</Default>
</Package>
```

Listing 5: Qt installscript.qs voor een zip

```
function Component()
{
    //Default component
        //Script van zip
}

Component.prototype.createOperations = function()
{
    // call default implementation to actually install
        ↪ README.txt!
    component.createOperations();

    if (systemInfo.productType === "windows") {
                component.addOperation("Execute"
                , "cmd"
                , "/c"
                , "C:\\\"Program Files\"\\WinRAR\\WinRAR.
                    ↪ exe"
                , "x"
                , "@TargetDir@\\pyusb-1.0.0a2.zip"
                , "@TargetDir@")
```

```
                    component.addOperation("Execute"
                    , "cmd"
                    , "/K"
                    , "\"cd"
                    , "@TargetDir@\\pyusb-1.0.0a2"
                    , "&&"
                    , "C:\\Python27\\python.exe"
                    , "setup.py"
                    , "install"
                    , "&&"
                    , "exit\"")
            }
}
```

Listing 6: NSIS nsi bestand

```
# define installer name
OutFile "installer0.2.0.exe"

# set desktop as install directory
InstallDir $PROFILE\NsisExample

# default section start
Section

# define output path
SetOutPath $INSTDIR

# specify file to go in output path
File python-2.7.3.msi
File pyusb-1.0.0a2.zip

# define uninstaller name
WriteUninstaller $INSTDIR\uninstaller.exe


#-------
# default section end
SectionEnd

Section

ExecWait '"msiexec" /i "$INSTDIR\python-2.7.3.msi" /quiet
    ↪ '

SectionEnd
```

```
Section

ZipDLL::extractall "$INSTDIR\pyusb-1.0.0a2.zip" "$INSTDIR
    ↪ "

ExecWait 'cmd /K "cd "$INSTDIR\pyusb-1.0.0a2" && "C:\
    ↪ Python27\python.exe" "setup.py" install && exit"'

SectionEnd

# create a section to define what the uninstaller does.
# the section will always be named "Uninstall"
Section "Uninstall"

ExecWait '"msiexec" /i "$INSTDIR\python-2.7.3.msi"'

# Always delete uninstaller first
Delete $INSTDIR\uninstaller.exe

# now delete installed file
Delete $INSTDIR\python-2.7.3.msi
Delete $INSTDIR\pyusb-1.0.0a2.zip

RMDir /r $INSTDIR\pyusb-1.0.0a2

SectionEnd
```

Listing 7: WiX wxs bestand

```
<?xml version="1.0"?>
<Wix xmlns="http://schemas.microsoft.com/wix/2006/wi">
   <Product Id="*" UpgradeCode
       ↪ ="12345678-1234-1234-1234-111111111111"
           Name="Python Framework Installer" Version
               ↪ ="0.0.1" Manufacturer="PJ Industries"
               ↪ Language="1033">
       <Package InstallerVersion="200" Compressed="yes"
           ↪ Comments="Windows Installer Package"/>
       <Media Id="1" Cabinet="product.cab" EmbedCab="yes
           ↪ "/>

       <Directory Id="TARGETDIR" Name="SourceDir">
          <Directory Id="PersonalFolder">
             <Directory Id="INSTALLDIR" Name="Example">
                <Component Id="ApplicationFiles" Guid
```

```
    ↪ ="12345678-1234-1234-1234-222222222222">
    ↪
                        <RegistryKey Root
                            ↪ ="HKCU"
                                                Key
                                                    ↪ ="
                                                    ↪ Software
                                                    ↪ \
                                                    ↪ My
                                                    ↪
                                                    ↪ Applicati
                                                    ↪ \
                                                    ↪ Uninstall
                                                    ↪ ">
                                                    ↪
                                    <
                                        ↪ RegistryValue
                                        ↪
                                        ↪ Value
                                        ↪ ="
                                        ↪ RobMen
                                        ↪
                                        ↪ Was
                                        ↪
                                        ↪ Here
                                        ↪ ."
                                                            Type
                                                                ↪
                                                                ↪
                                                                ↪
                                                                ↪
                                                            KeyPa
                                                                ↪
                                                                ↪
                                                                ↪
                                                                ↪
                                                                ↪
                                                                ↪
                                                </RegistryKey>
<File Id="ApplicationFile1" Source="
    ↪ python-2.7.3.msi"/>
                                <File Id="
                                    ↪ ApplicationFile2"
                                    ↪  Source="pyusb
                                    ↪ -1.0.0a2.zip"/>
                                <RemoveFolder Id='
```

```
                                        ↪ INSTALLDIR' On='
                                        ↪ uninstall' />
                </Component>
            </Directory>
        </Directory>
    </Directory>

    <Feature Id="DefaultFeature" Level="1">
        <ComponentRef Id="ApplicationFiles"/>
    </Feature>

    <InstallExecuteSequence>
        <Custom Action='FooAction' After='InstallFiles
            ↪ '/>
    </InstallExecuteSequence>
  </Product>
</Wix>
```

Referenties

[1] Chocolatey Documentation. `https://chocolatey.org/docs`, 2016. [Online; geraadpleegd 3-08-2016].

[2] Chocolatey Main page. `https://chocolatey.org/`, 2016. [Online; geraadpleegd 3-08-2016].

[3] esky Main Page. `https://pypi.python.org/pypi/esky`, 2016. [Online; geraadpleegd 5-08-2016].

[4] Google Omaha Main Page. `https://github.com/google/omaha`, 2016. [Online; geraadpleegd 5-08-2016].

[5] Inno Setup Main Page. `http://www.jrsoftware.org/isinfo.php`, 2016. [Online; geraadpleegd 5-08-2016].

[6] NSIS Features. `http://nsis.sourceforge.net/Features`, 2016. [Online; geraadpleegd 2-08-2016].

[7] NSIS Main page. `http://nsis.sourceforge.net/Main_Page`, 2016. [Online; geraadpleegd 3-08-2016].

[8] QT Installer Framework Documentation. `http://doc.qt.io/qtinstallerframework/`, 2016. [Online; geraadpleegd 4-08-2016].

[9] QT Installer Framework Documentation Scripting. `http://doc.qt.io/qtinstallerframework/scripting.html`, 2016. [Online; geraadpleegd 5-08-2016].

[10] QT Installer Framework Overview. http://doc.qt.io/qtinstallerframework/ifw-overview.html, 2016. [Online; geraadpleegd 5-08-2016].

[11] WinSparkle Documentation. https://github.com/vslavik/winsparkle/wiki, 2016. [Online; geraadpleegd 5-08-2016].

[12] WinSparkle Main page. https://winsparkle.org/, 2016. [Online; geraadpleegd 5-08-2016].

[13] WinSparkle User-Experience. https://github.com/vslavik/winsparkle/wiki/User-Experience, 2016. [Online; geraadpleegd 5-08-2016].

[14] WiX Toolset Main Page. http://wixtoolset.org/, 2016. [Online; geraadpleegd 3-08-2016].

[15] WiX Toolset Merge Modules. https://www.firegiant.com/wix/tutorial/upgrades-and-modularization/mergers/, 2016. [Online; geraadpleegd 3-08-2016].