TUM Open Infra Platform

Developer Documentation

Content

[About this Document 3](#_Toc465084389)

[Name conventions 3](#_Toc465084390)

[Getting started 3](#_Toc465084391)

[Needed Software 3](#_Toc465084392)

[Long time support 5](#_Toc465084393)

[Clone the repository 5](#_Toc465084394)

[Compiling 5](#_Toc465084395)

[Creating a Visual Studio 2015 x64 Solution 5](#_Toc465084396)

[Prebuild early bindings 13](#_Toc465084397)

[After compiling 14](#_Toc465084398)

[Workflow: How to commit code? 15](#_Toc465084399)

[Contacts 19](#_Toc465084400)

[The Developers 19](#_Toc465084401)

[Website 19](#_Toc465084402)

[TUM Open Infra Platform 21](#_Toc465084403)

[Use Cases 21](#_Toc465084404)

[Use Case #1: Viewing alignment, road and terrain data 21](#_Toc465084405)

[Use Case #2: Import/Export LandXML, OKSTRA and IFC Alignment data 21](#_Toc465084406)

[Use Case #3: Create new alignment and terrain data 21](#_Toc465084407)

[Features 21](#_Toc465084408)

[Background knowledge 22](#_Toc465084409)

[Alignment 22](#_Toc465084410)

[Horizontal Alignment 22](#_Toc465084411)

[LandXML 23](#_Toc465084412)

[File format issues 23](#_Toc465084413)

[Clothoid 24](#_Toc465084414)

[Clothoid implementation 24](#_Toc465084415)

[Vertical alignment 27](#_Toc465084416)

[Computation of vertical alignments (parabolas) 27](#_Toc465084417)

[Vertical alignment that starts with a parabola 28](#_Toc465084418)

[IFC Alignment 28](#_Toc465084419)

[RoadXML 29](#_Toc465084420)

[Tools 30](#_Toc465084421)

[JSDAI 30](#_Toc465084422)

[AutoCAD Civil 3D 2014 31](#_Toc465084423)

[Code Guidelines 32](#_Toc465084424)

[Syntax #1 32](#_Toc465084425)

[Ident style #1 32](#_Toc465084426)

[Comments #1 33](#_Toc465084427)

[Comments #2 33](#_Toc465084428)

[Tags 33](#_Toc465084429)

[Includes #1 33](#_Toc465084430)

[Includes #2 33](#_Toc465084431)

[CMake 34](#_Toc465084432)

[Code Architecture 34](#_Toc465084433)

[Code in Detail 34](#_Toc465084434)

[IFC Eearly Binding Meta Template Library (ifcEMT) 34](#_Toc465084435)

[Testing 36](#_Toc465084436)

[Testdata 36](#_Toc465084437)

[Unit Testing 38](#_Toc465084438)

[IFC Alignment Common Mistakes 38](#_Toc465084439)

[Check #1 38](#_Toc465084440)

[Check #2 38](#_Toc465084441)

[Check #3 39](#_Toc465084442)

[Check #4 39](#_Toc465084443)

# About this Document

This short guide shows you how to compile the project TUM Open Infra Platform.

***Warning:*** This document is only snapshot of the current state and may be become obsolete in the future.

# Name conventions

The release title of the software is TUM Open Infra Platform. Sometimes it is only called Open Infra Platform or more shortly OIP.

# Getting started

## Clone the repository

The source code of Open Infra Platform is managed using the hg (aka Mercurial) version control system (get TortoiseHg from <http://tortoisehg.bitbucket.org/>). To obtain the source code follow the listed instructions:

1. Open a console window.
2. Switch to directory where you want to clone the repository.
3. Type hg clone <https://Vertexwahn@bitbucket.org/Vertexwahn/openinfraplatform>2

into a console window. If you are not familiar with the windows command line, there is also a visual way.

Contact [julian.amann@tum.de](mailto:julian.amann@tum.de) to get test data.

## Needed Software

To be able to build Open Infra Platform on your machine you need some tools and third party libraries.

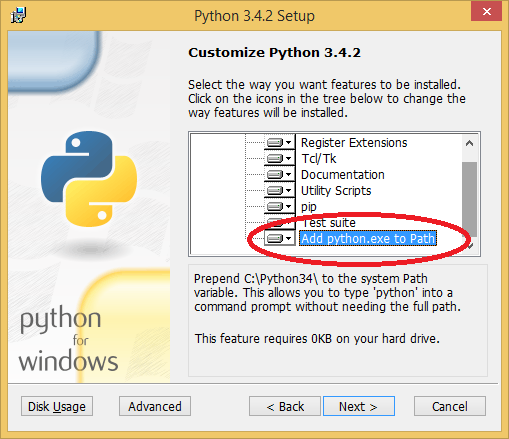
### Third Party Libraries

Install the following tools and libraries:

1. Visual Studio 2015 32 Bit (64 Bit version does not exist).
2. Windows Software Development Kit (SDK) for Windows 10 Version 10.0.14393.0. This can either be installed as standalone sdk or with the visual studio distribution. We recommend installing it as standalone software.

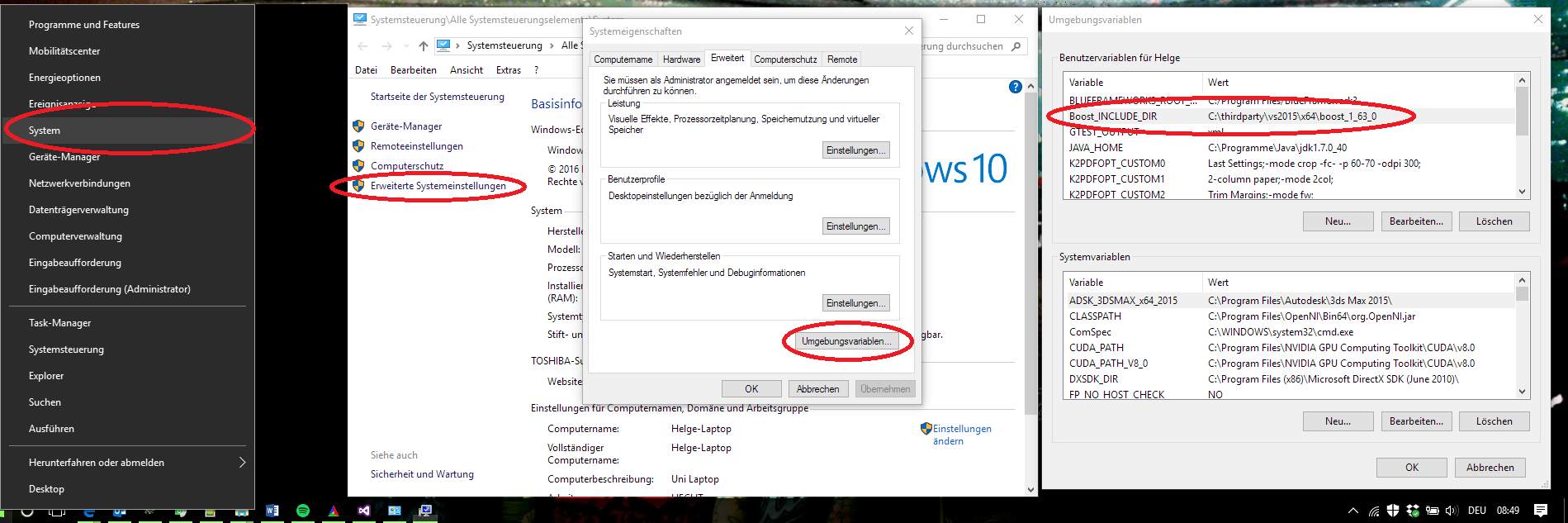
(<https://developer.microsoft.com/de-de/windows/downloads/windows-10-sdk>).

1. CMake 3.7.2 from <http://www.cmake.org/>
2. Qt 5.8.0 (<https://www.qt.io/download-open-source/#section-2>)
3. Python 3.5.1 **x64**: Add python.exe to Path. (Note: It is very important that an x64 Version of Python is used instead of an x86 – otherwise the compilation will not work. Please do not use any other version of python because this is untested – also install this python version before you build the boost libraries, if you build them yourself) (<https://www.python.org/downloads/>)



5.1 Alternatively you can also install the Anaconda Python distribution to get the full Unix python experience, including package management etc. (Downloads: <https://www.continuum.io/downloads> , Documentation: <https://docs.continuum.io/>). This also automatically sets all important environment variables.

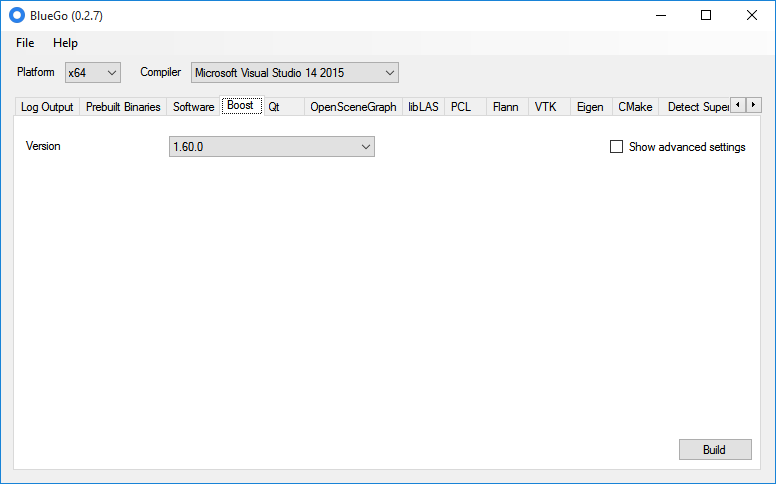
1. Visual Leak Detector v2.4rc2 (<https://vld.codeplex.com/releases>)
2. Boost 1.63.0 (<https://sourceforge.net/projects/boost/files/boost-binaries/>). Please install to “C:\thirdparty\vs2015\x64\boost\_1\_63\_0”. It can be useful to add this path into the environment variables.



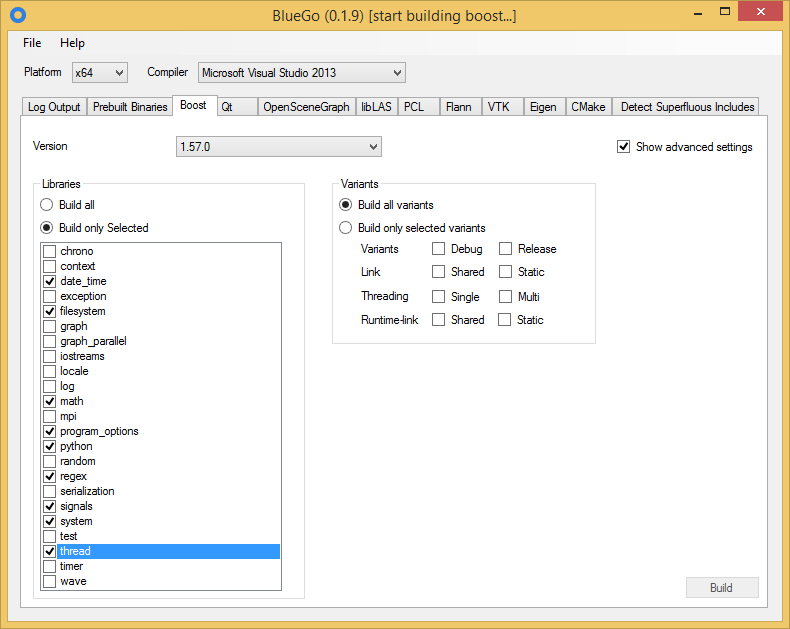
1. Eigen 3.3.3 (<http://eigen.tuxfamily.org/index.php?title=Main_Page>). Please install/unzip to “C:\thirdparty\vs2015\x64\Eigen\_3.3.3”. Since Eigen is a header only library, unzipping the contents of the downloaded zip serves as full installation. There are no setups involved.
2. BlueFramework 3 (<https://bitbucket.org/tumcms/blueframework3-public/downloads>). You can also build the BlueFramework3 yourself (not recommended for most users). Similar to boost, it is useful to set the “BLUEFRAMEWORK3\_ROOT\_DIR” environment variable to the location where your BlueFramework is installed (“C:\Program Files\BlueFramework3”).

### Building Boost Yourself

The current version of BlueGo can be downloaded from <https://bitbucket.org/Vertexwahn/bluego/downloads/>. Use the following settings in BlueGo to build boost:

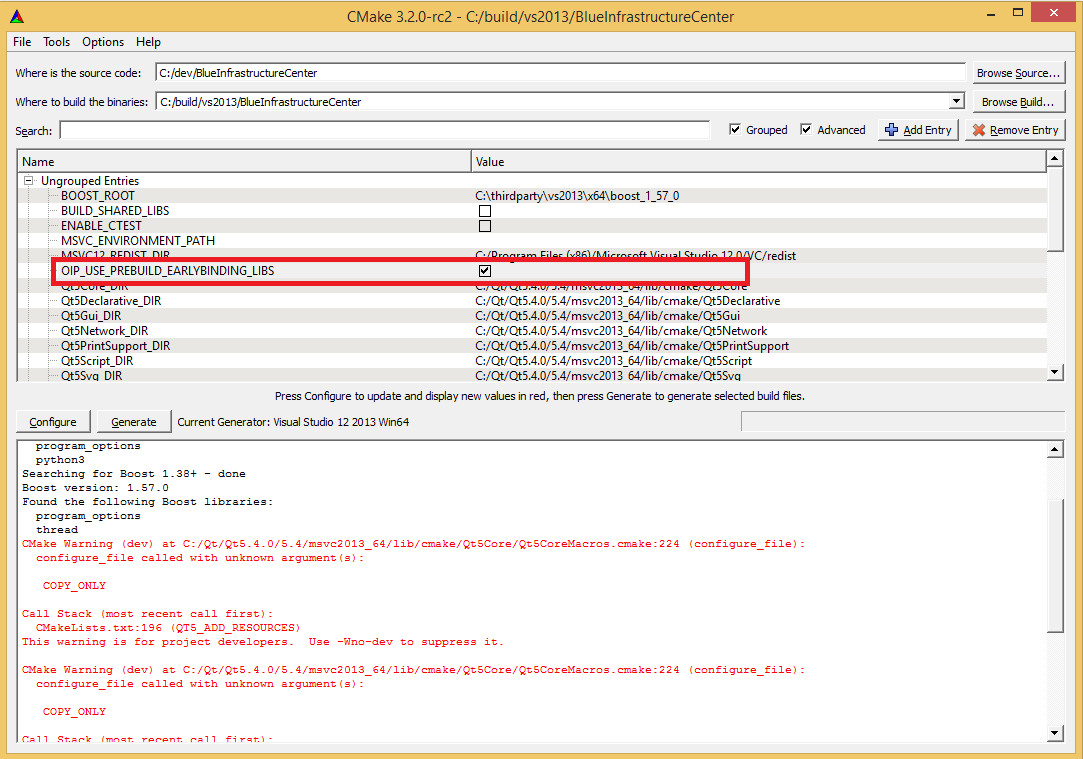


Actually not all libraries of boost are needed – the following settings for a reduced build should also work:



### Prebuild early bindings

Prebuild early bindings can be downloaded via the batch script “download\_prebuild\_libs.cmd” which can be found in the bootstrap folder (“<OIP source dir>\bootstrap”). This will create a folder “prebuild\_libs” in the “OpenInfraPlatform” directory. If you like to use prebuild early binding libs you also need to enable the OIP\_USE\_PREBUILD\_EARLYBINDING\_LIBS check box in CMake:

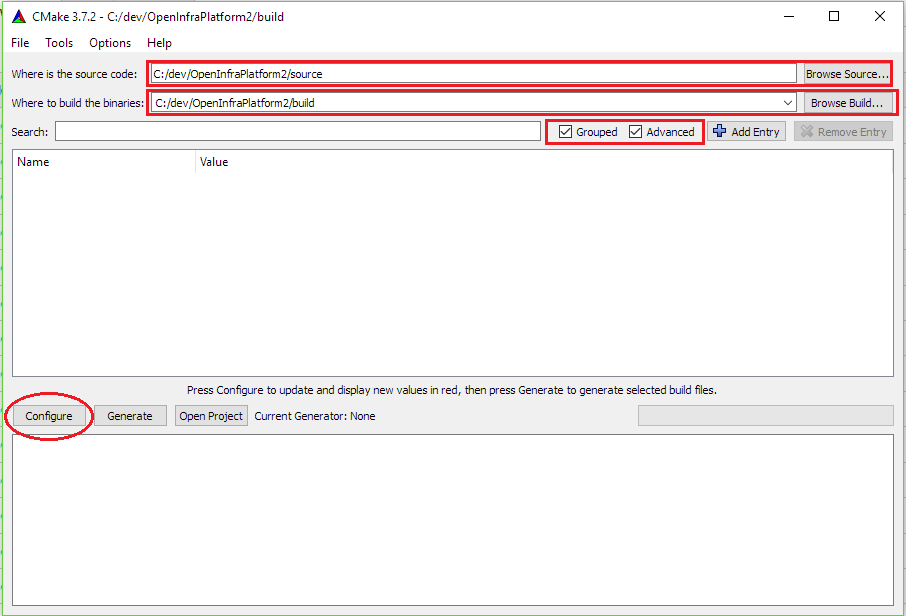


Since the prebuild libraries do not change very often the need only compiled once. Also building the early binding takes some time that can be saved using this approach.

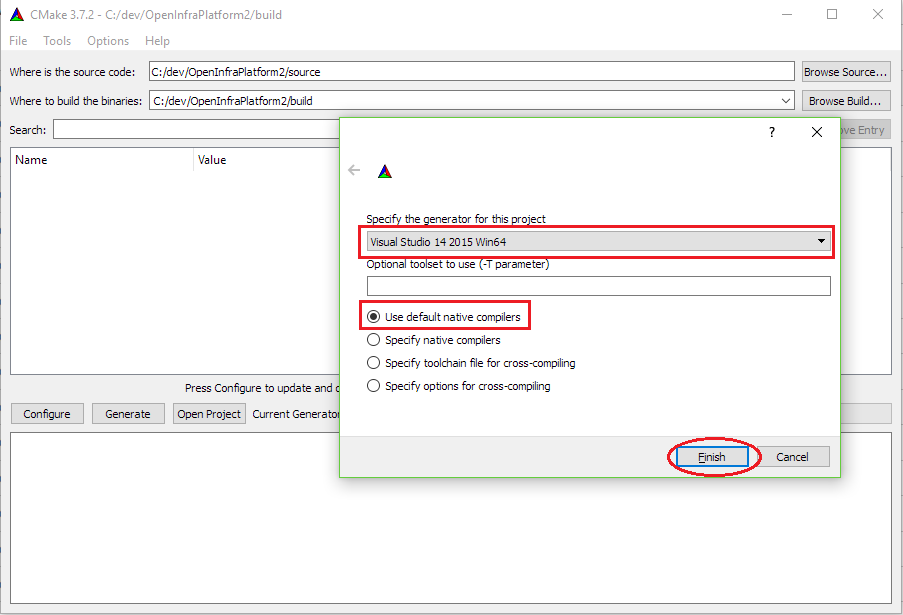
## Compiling

### Creating the Solution

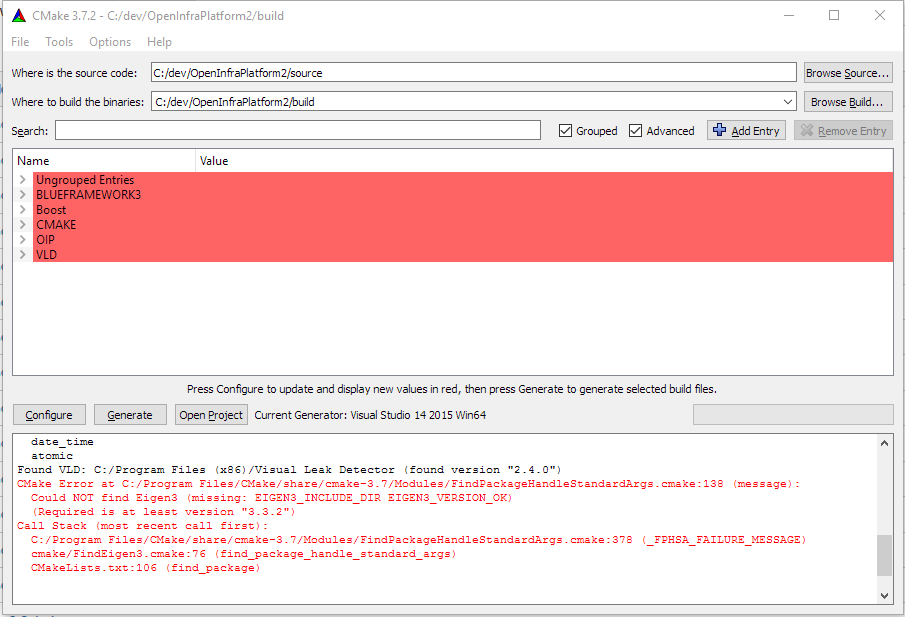
Open CMake and set the text of the “Where is the source code:” text box to path where the source code of the OpenInfraPlatform2 repository is located. Set the “Where to build the binaries:” directory to where you want to build the project (this path does not need to exist).



Then click the configure button. Then select Visual Studio 14 2015 Win64 as the version and choose the default native compilers. Then hit finish.



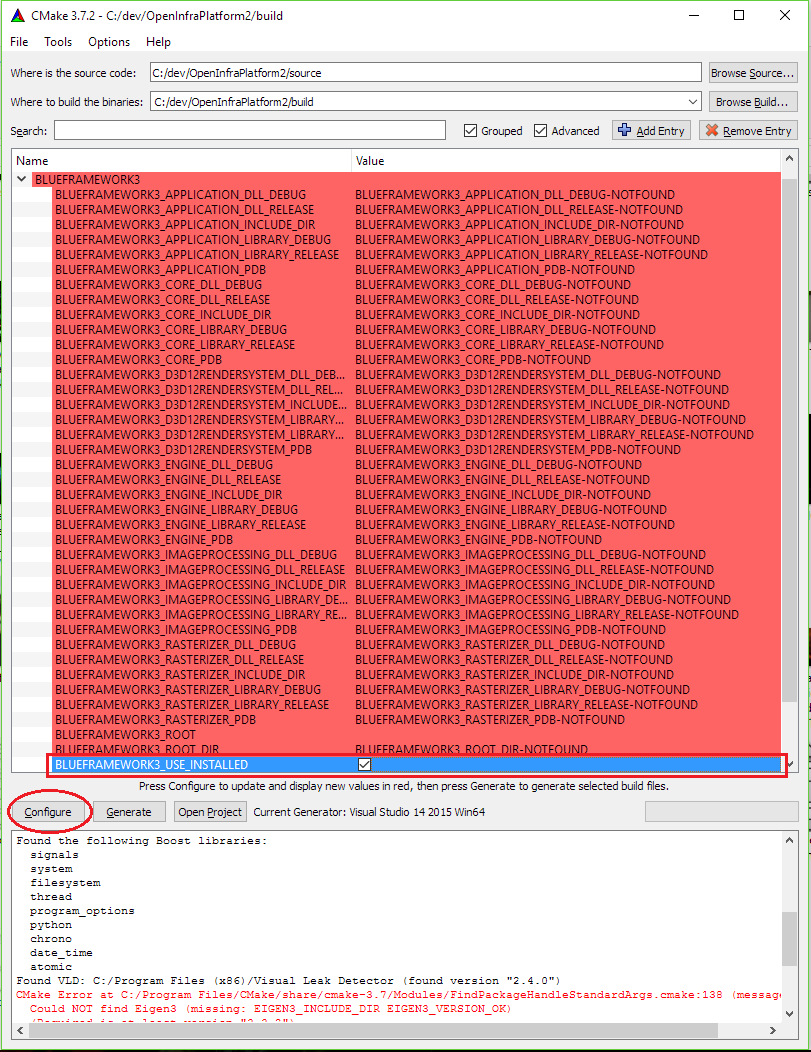
Then CMake should look for the first libraries. After some time, it should stop and look similar to this:



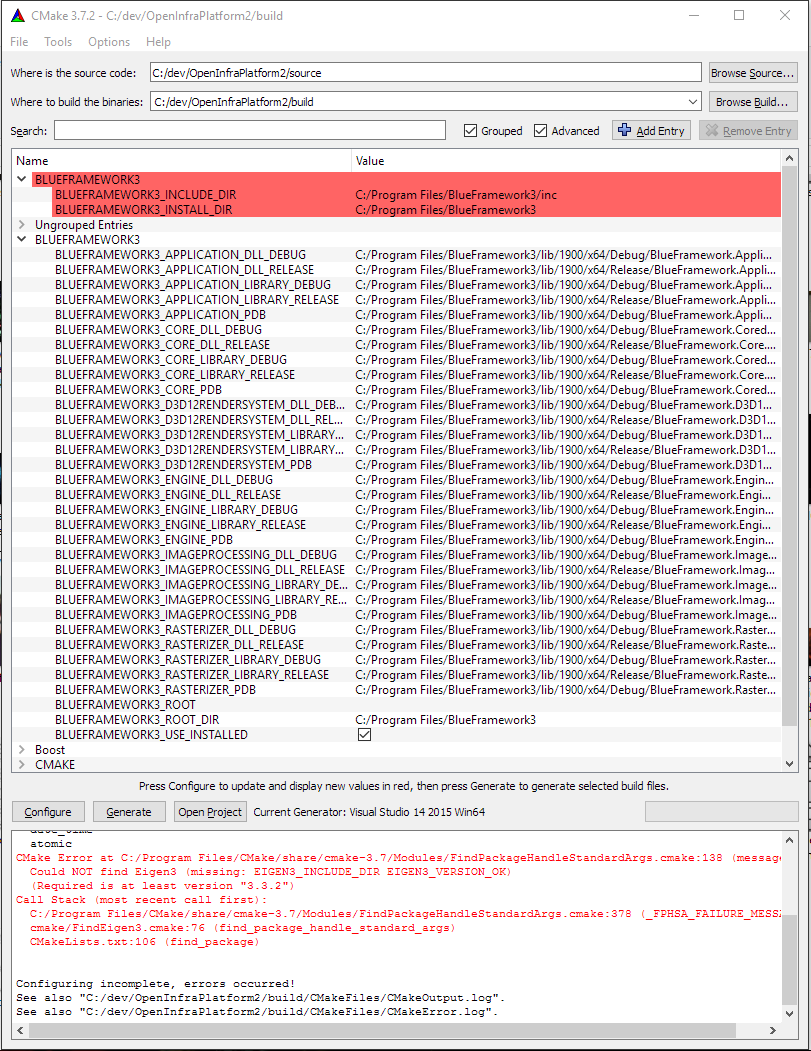
Depending on the environment variables etc., CMake will find some libraries and programs immediately while some will require some help to locate.

#### Finding the BlueFramework3

If you installed the BlueFramework with the binary installer, you can simply click the checkbox in the line saying BLUEFRAMEWORK\_USE\_INSTALLED.



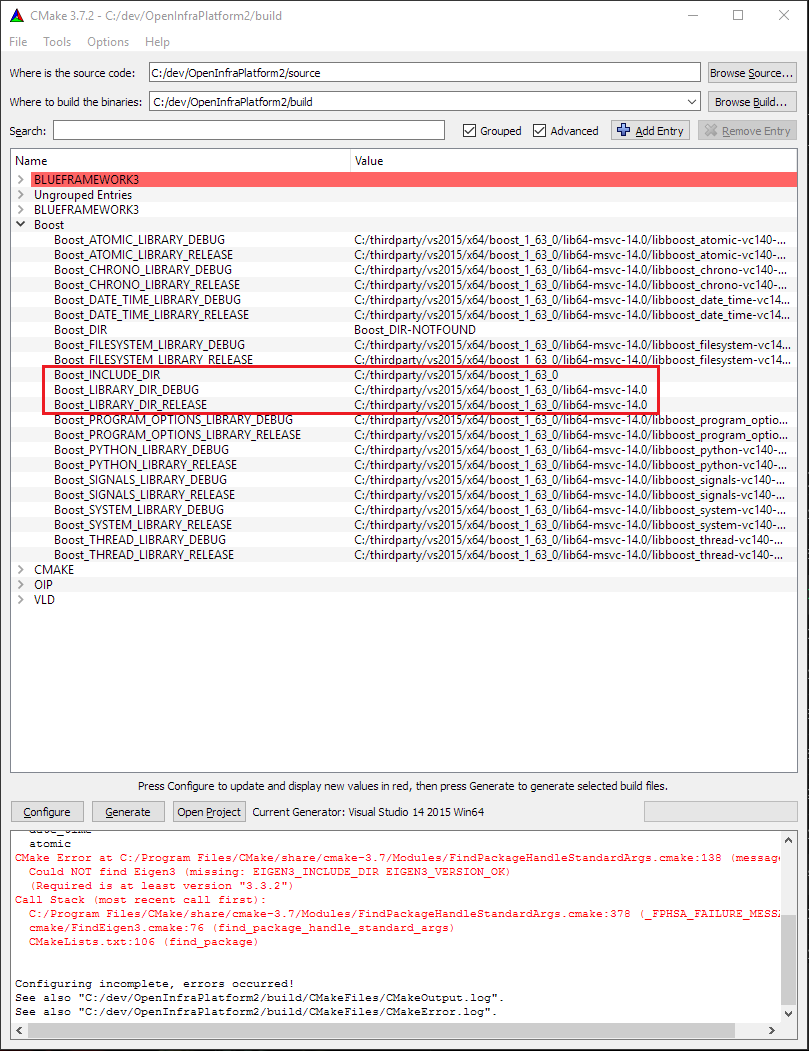
It should then be able to locate the BlueFramework libraries after hitting configure again.



If you didn’t use the binary installer, set the BLUEFRAMEWORK\_ROOT\_DIR to the root directory of the BlueFramework. It should then also automatically detect the libraries.

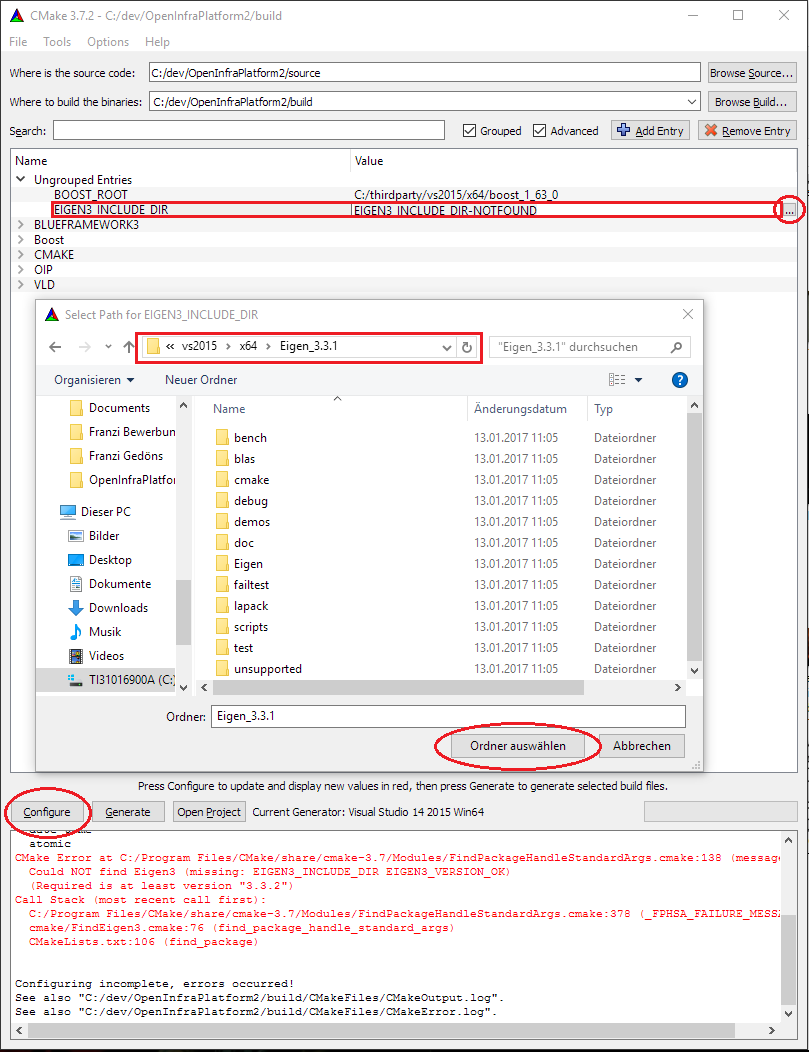
#### Finding Boost

If you have set the Boost\_INCLUDE\_DIR to the boost root dir in your environment variables, it should automatically detect the libraries. If not, set the Boost\_LIBRARY\_DIR\_DEBUG and RELEASE paths as following:

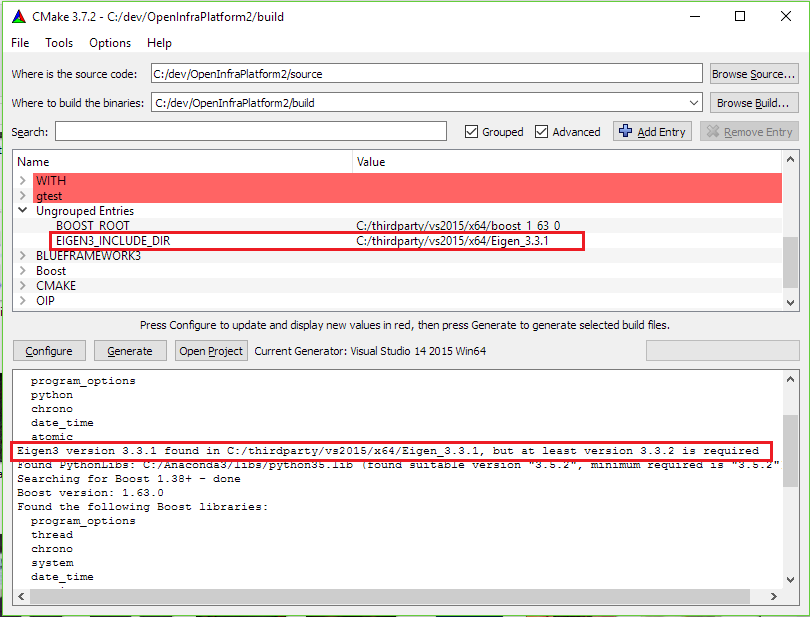


#### Finding Eigen

To find eigen, set the EIGEN3\_INCLUDE\_DIR path to the root path of your Eigen installation. Just click the “Browse” button next to the path and navigate to your installation directory. Then select this folder and click the configure button again.

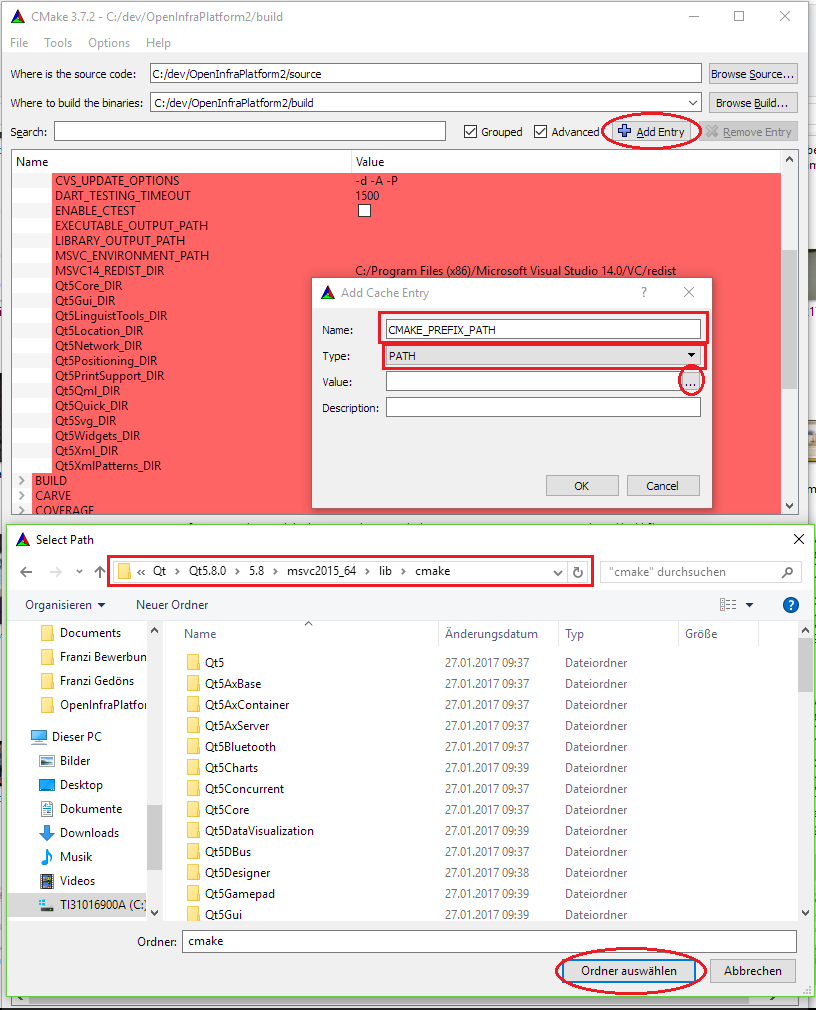


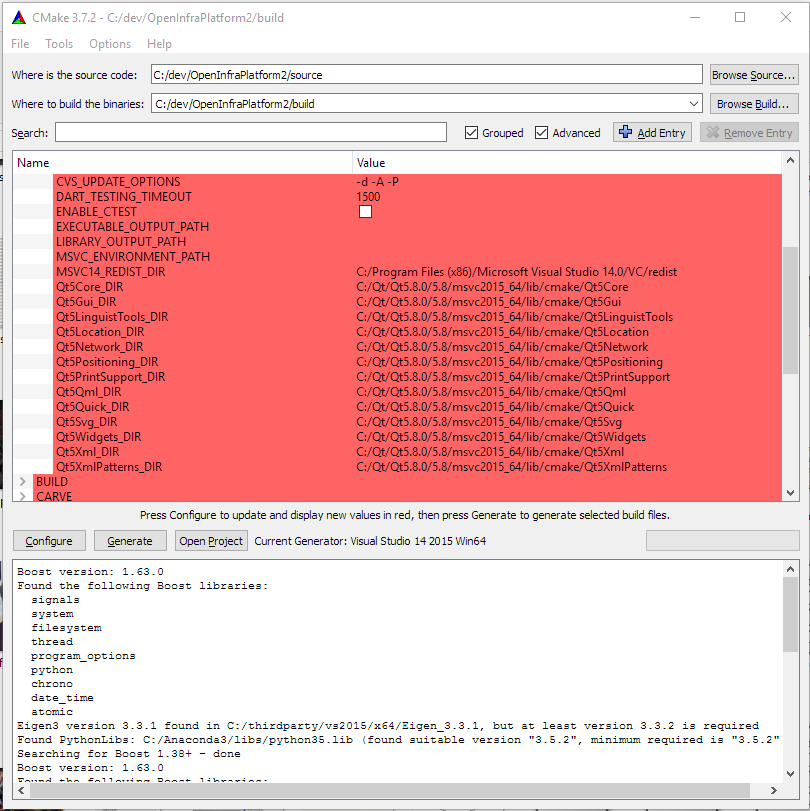
After that, there should be a line in the log saying that Eigen3 has been found. Even though Eigen 3.3.2 is required, Eigen 3.3.1 works just fine.



#### Finding Qt

To find Qt fast, it is best the add the entry “CMAKE\_PREFIX\_PATH” with the path to “C:\Qt\Qt5.8.0\5.8\msvc2015\_64\lib\cmake” to either the environment variables or in the CMake GUI. This can be done by clicking the “Add Entry” button in the GUI and then selecting PATH as Type and navigating to the path in the value field. CMake should then be able to locate the Qt libraries automatically.





#### Finding the rest

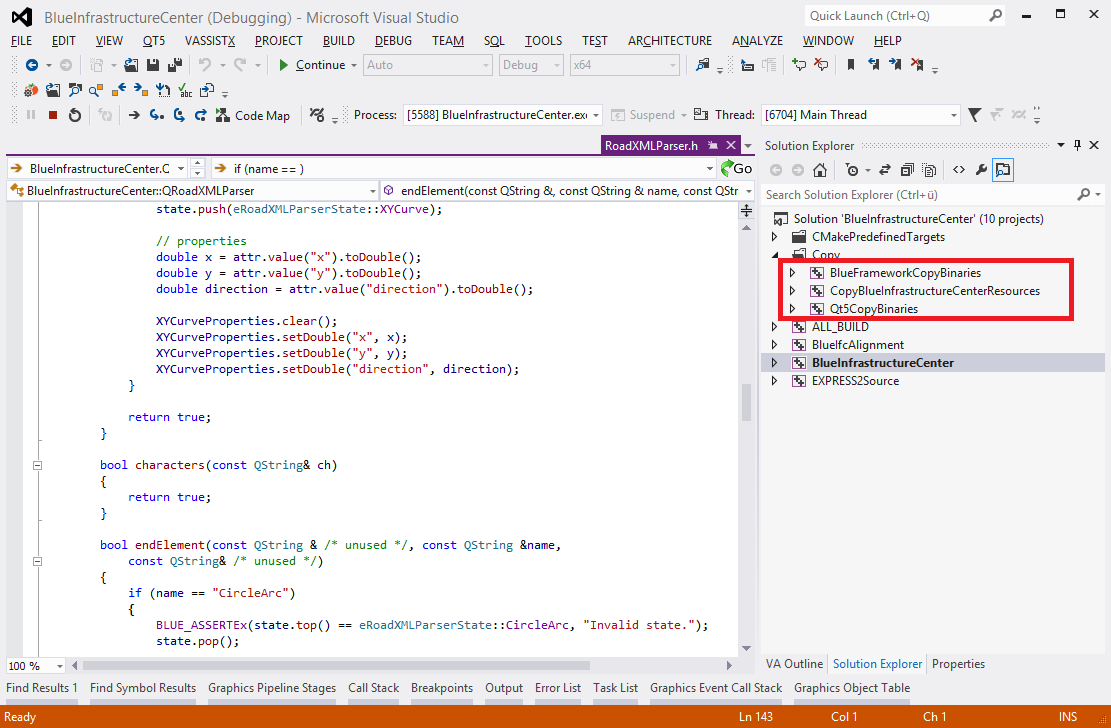
All other libraries should be found automatically, like Python and VLD etc.

### Compiling the Solution

When opening the generated solution by clicking OpenProject in the CMake GUI the first time the ALL\_BUILD target is selected. Swich this to OpenInfraPlatform.UI as StartUp project.

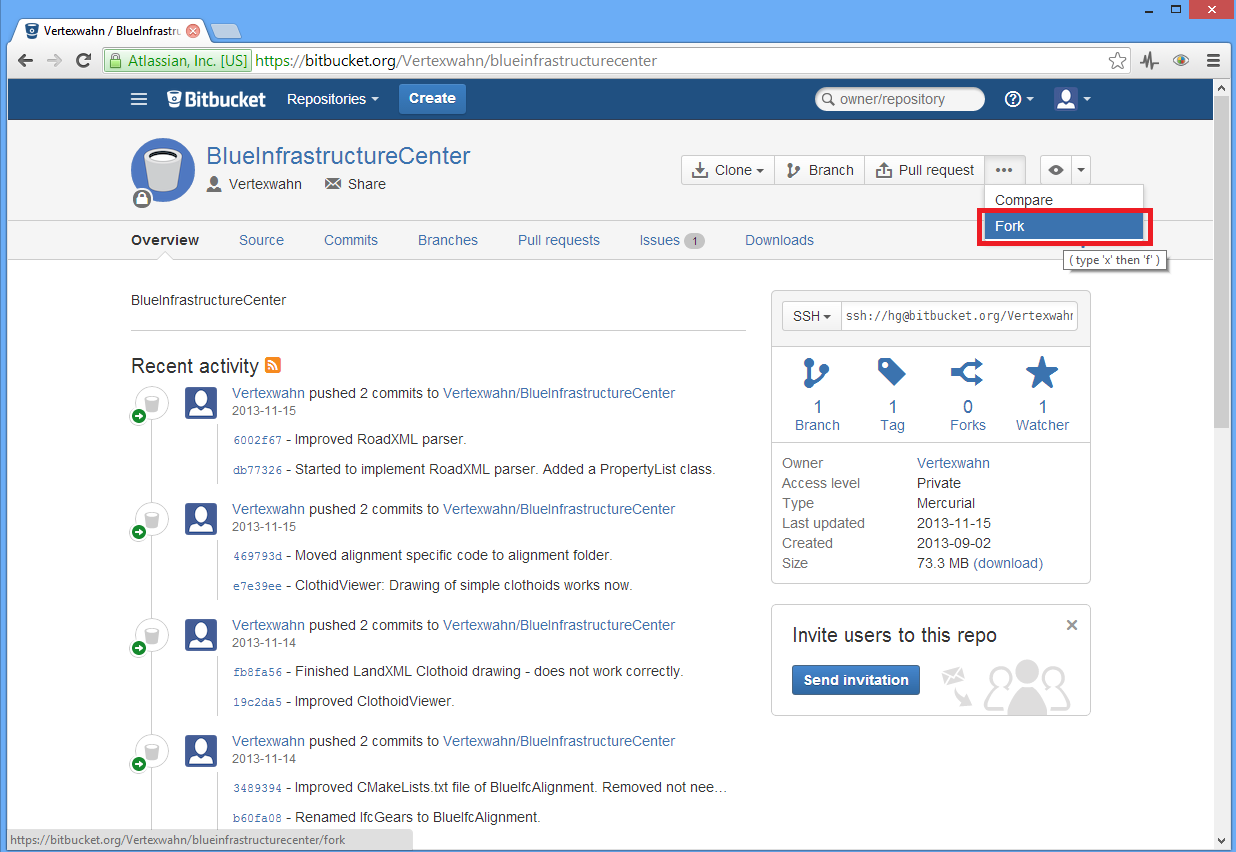
## After compiling

After successful compiling you can run the program. When running OpenInfraPlatform.UI the first time there will be an error message that there are missing DLL files (from Qt and BlueFramework). You can either copy these missing files by hand (which is not suggested) or using the generated projects BlueFrameworkCopyBinaries which copies the BlueFramework DLLs and Qt5CopyBinaries which copies the Qt 5DLLs to the right place. Additionally you should run the CopyOpenInfraPlatformUIResources project which copies needed resources to the right place.

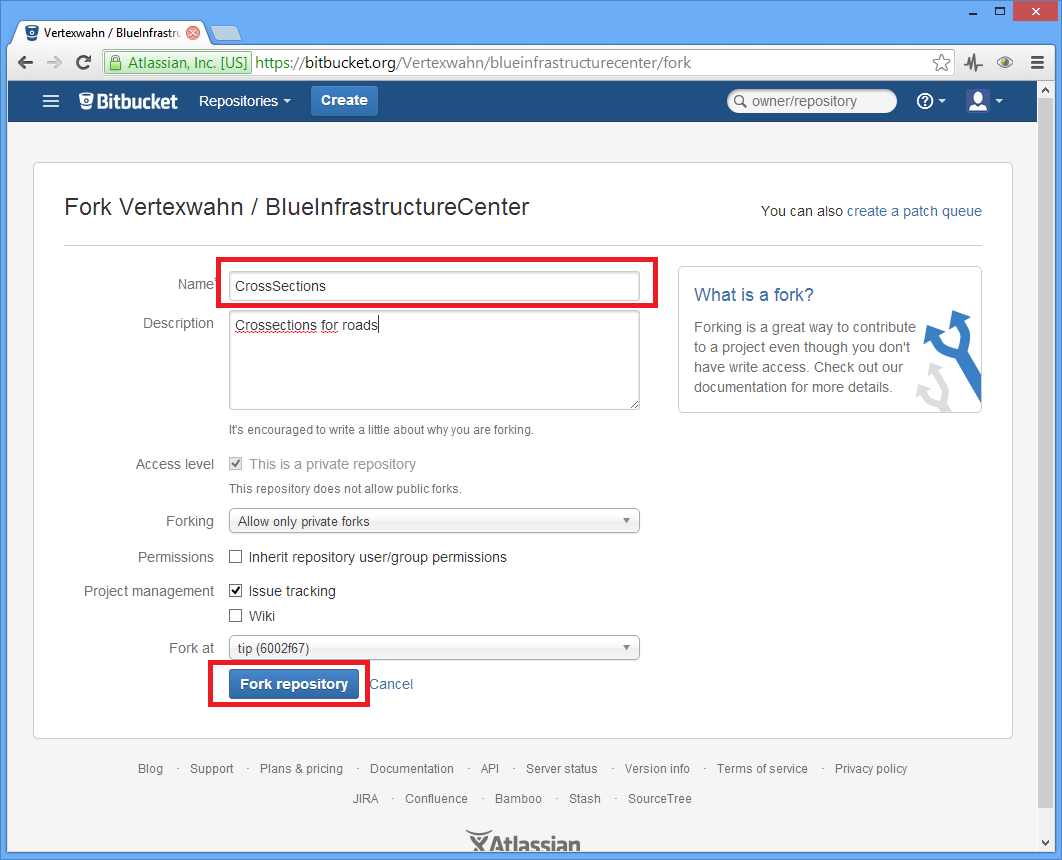


## Workflow: How to commit code?

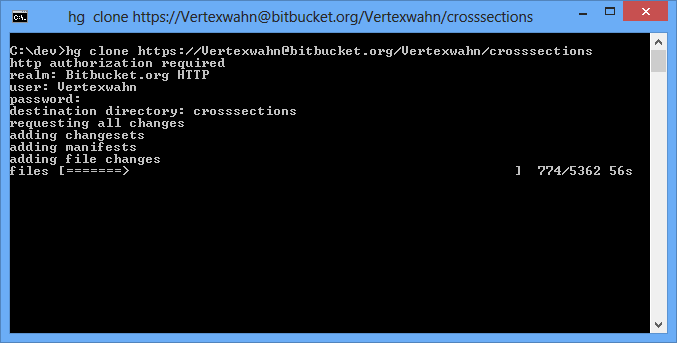
Create a fork of the OpenInfraPlatform project. This needs to be done, because of the access management policy. You are only allowed to read from the repository not to write to it. For that reason you have to make an own fork. Your forked repository should be private.



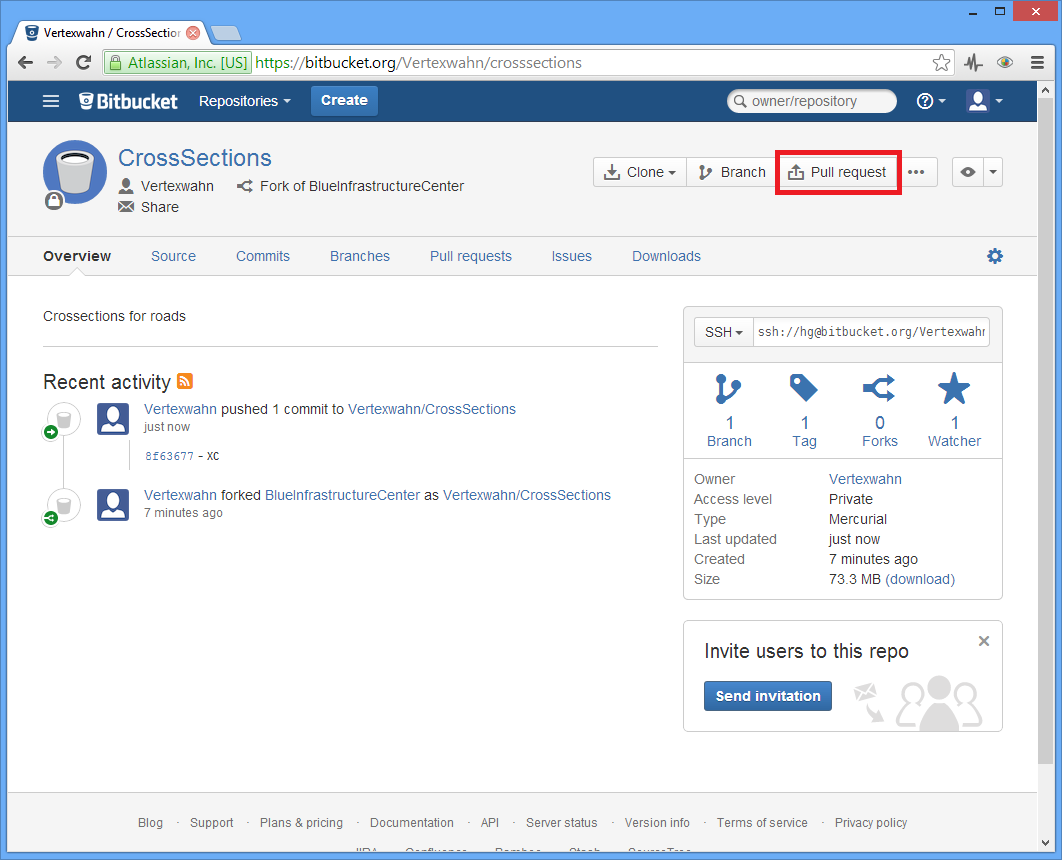
You can name your fork as you wish, but a good name of the fork is a word that describes what you are working on in this fork.



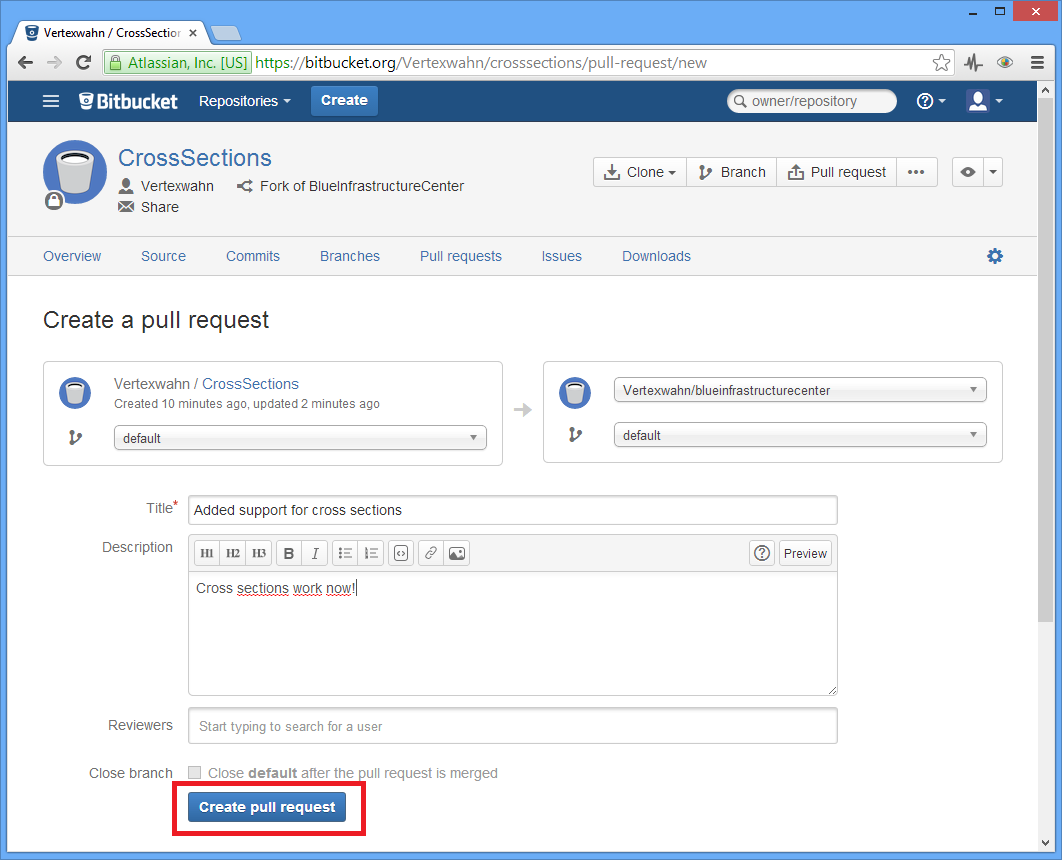
After creating the fork you should clone your own fork. For instance clone it by typing something like “hg clone <https://Vertexwahn@bitbucket.org/Vertexwahn/crosssections>” to your console.



After cloning the forked repository you should only make changes to your forked repository. You can commit and push changes to your forked repository. If you are done with your task you can make a “Pull request” to the original OpenInfraPlatform repository.

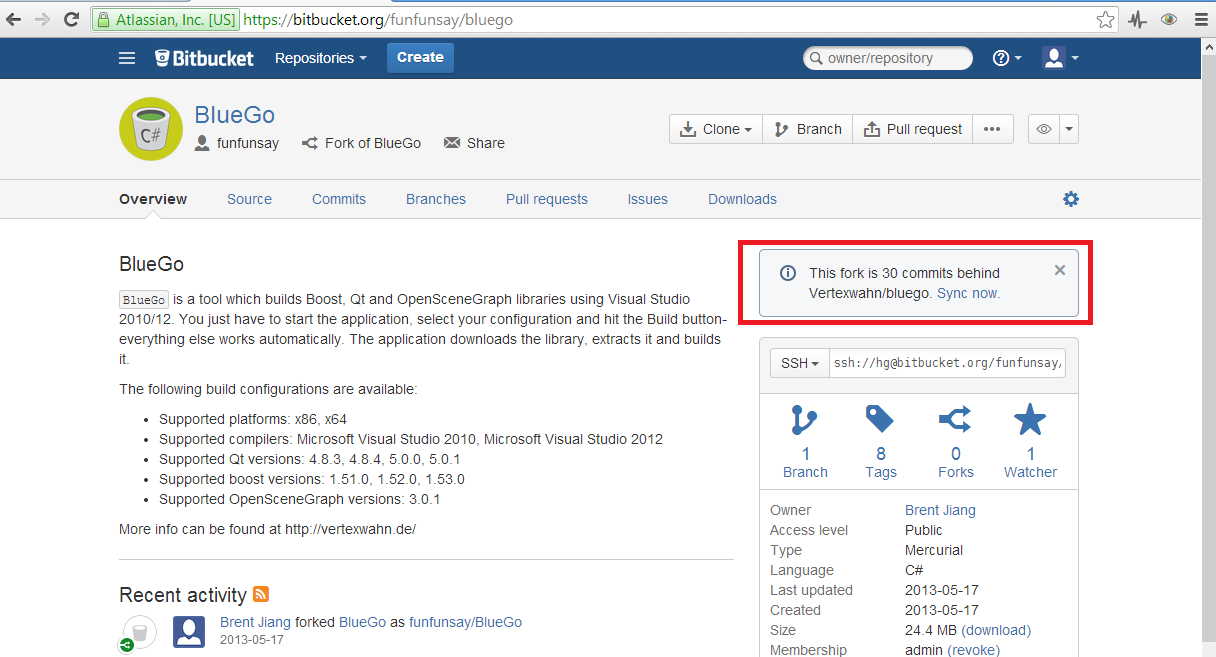


The pull request will be sent to the project manager:

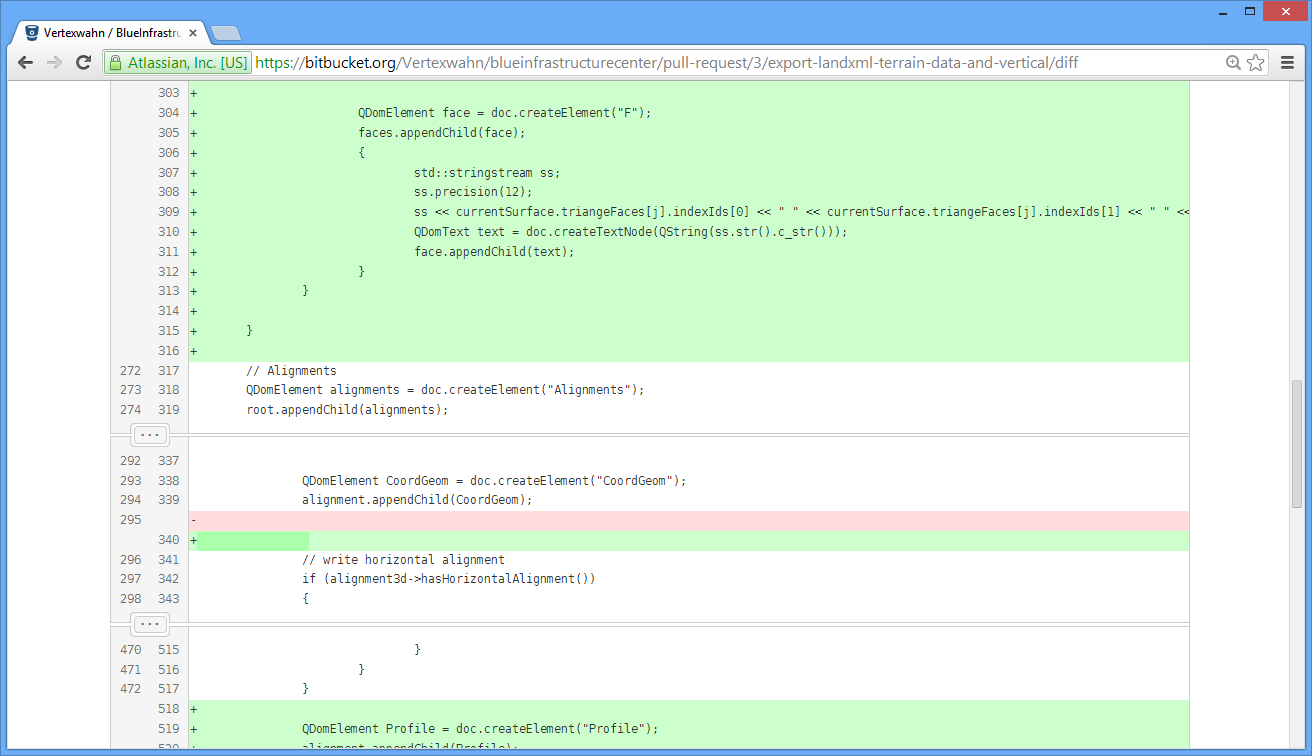


Then a code review will take place. The project manager will inform you what needs to be changed before the pull request is accepted. Once the code review is finished your changes will be applied to the original OpenInfraPlatform.

Sometimes there can be changes in the original repository, that you want also have in your forked repository. To get the change you have to Sync your forked repository by clicking “Sync now”.



This workflow helps the project manager to see what changes have been introduced in the code base. Also it is possible via BitBucket to add comment to your changes.



# Contacts

## The Developers

**Project Managers:**  
Julian Amann aka Vertexwahn ([julian.amann@tum.de](mailto:julian.amann@tum.de); [vertexwahn@gmx.de](mailto:vertexwahn@gmx.de))

**Current Development Team**Fabian Schöttl ([fabi-schoettl@hotmail.de](mailto:fabi-schoettl@hotmail.de))  
Helge Hecht ([helge.hecht@tum.de](mailto:helge.hecht@tum.de))

**Former Lead Test Engineer/Lead User Experience Engineer:**   
Javier Jubierre ([jubierre@bv.tum.de](mailto:jubierre@bv.tum.de))

**Former project managers:**

Dominic Singer ([dominic.singer@tum.de](mailto:dominic.singer@tum.de))Fabian Ritter ([mail@redinkinc.de](mailto:mail@redinkinc.de))

**Former developers:**Alexander Widner ([ga96heq@mytum.de](mailto:ga96heq@mytum.de); [alexander.widner@rwth-aachen.de](mailto:alexander.widner@rwth-aachen.de))  
Michael Kern aka mchke89 ([kernm@in.tum.de](mailto:kernm@in.tum.de))  
Daniel Below aka Ancaria ([daniel.below@tum.de](mailto:daniel.below@tum.de); [dbelow@hotmail.de](mailto:dbelow@hotmail.de))  
Patrick Geisler ([geisler.patrick@gmx.de](mailto:geisler.patrick@gmx.de))Nishant Gupta ([nishant.gupta@tum.de](mailto:nishant.gupta@tum.de), [nishant8568@gmail.com](mailto:nishant8568@gmail.com))  
Ahmed Mustafa ([ahmed.mustafa@tum.de](mailto:ahmed.mustafa@tum.de))

# Website

The official website can be found here:

<http://www.cms.bgu.tum.de/oip> (redirects to <https://www.cms.bgu.tum.de/de/forschung/projekte/oip>)

Also

<https://www.cms.bgu.tum.de/de/forschung/projekte/31-forschung/projekte/397-tum-open-infra-platform>

works, but is very ugly.

Short URL: <http://go.tum.de/309508>

We should try to get a better/easier URL such as <http://oip.cms.bgu.tum.de> or http://www.oip.de

This Website is for End Users of Open Infra Platform.

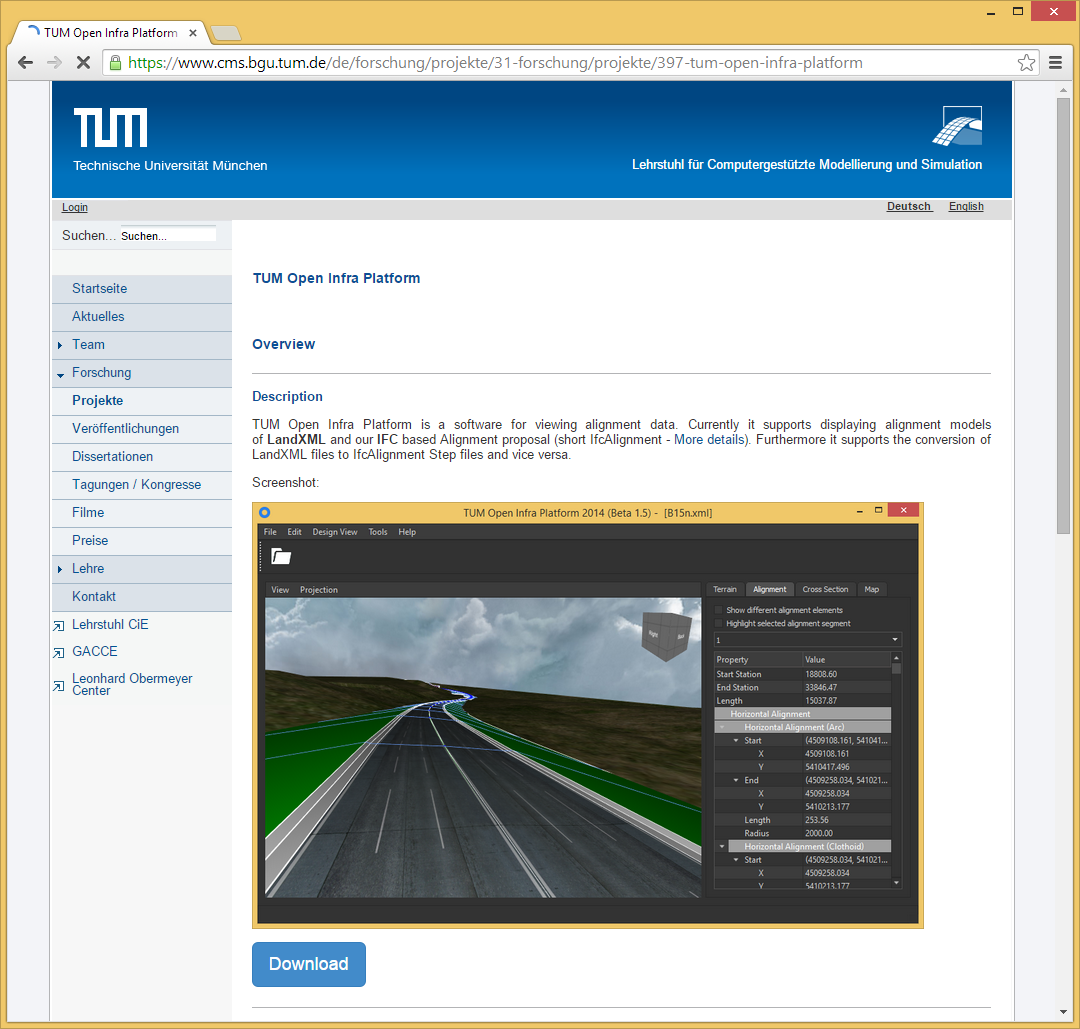


Figure 1: Website of TUM Open Infra Platform

**QR code for website:**



# TUM Open Infra Platform

## Use Cases

The following use cases should give you a small preview of the feature set of OIP.

### Use Case #1: Viewing alignment, road and terrain data

### Use Case #2: Import/Export LandXML, OKSTRA and IFC Alignment data

**Description:** A user wants to view LandXML Data.   
**Goal:** It should be possible to display LandXML data, including alignment data, digital elevation model and cross sections.  
**Actors:** Civil engineer.  
**Stakeholders:**

1. User wants to view the LandXML data in a graphical representation.

**Basic Course:**

1. We have an existing LandXML file that contains alignment data and a digital elevation model
2. A user selects the corresponding LandXML file and view it via the application

### Use Case #3: Create new alignment and terrain data

**Description:** A user wants to create an alignment and store it as an LandXML file or a user wants to import a XYZ file

## Features

Feature: IFC Alignment Export

Feature: LandXML Import

Feature: Excel Export

Feature: IFC Alignment Import

Feature: OKSTRA Import

Feature: OKSTRA Export

Feature: IFC Alignment Export

Feature: IFC Alignment Import

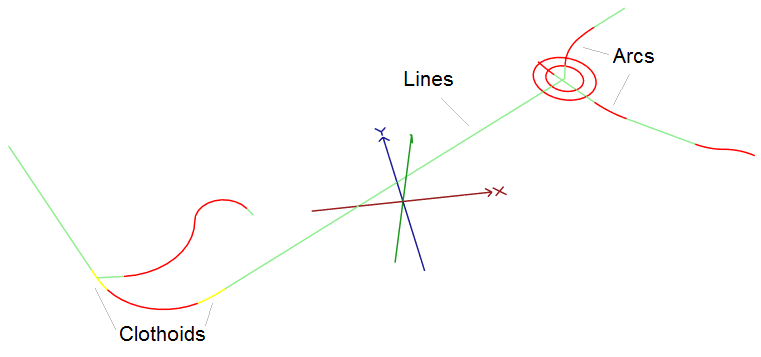
# Background knowledge

## Alignment

The core of each infrastructure project is its alignment. In this section different existing product models for alignment data are investigated.

## Horizontal Alignment

The horizontal alignment consists of lines, arcs and clothoids.



A line segment can be described by different properties. For instance line properties could be a start point, end point, length, direction, etc. The same is true for arcs and transition curves like clothoids.

|  |  |
| --- | --- |
| Semantic Property Identifier & type | Description |
| Line.Start : vector2d | Start point (x,y) of a line segement. |
| Line.End : vector2d | End point (x,y) of a line segement. |
| Line.Direction: vector2d | Direction of a line segment (Direction = End – Start) |
| Line.Length : double | Length of a line segement. |
| Arc.Center : vector2d |  |
| Arc.Start : vector2d |  |
| Arc.End : vector2d |  |
| Arc.Radius : double |  |
| Arc.Curvature : double |  |
| Arc.Length : double |  |
| Arc.Clockwise : bool |  |
| Arc.counterclockwise : bool |  |
| Arc.direction : double |  |
| Clothoid.StartRadius : double |  |
| Clothoid.EndRadius : double |  |
| Clothoid.StartCurvature : double |  |
| Clothoid.EndCurvature : double | End curvature of clothoid. |
| Clothoid.Lenght : double | Length of clothoid segment. |
| Clothoid.Constant : double | Clothoid constant value. |
| Clothoid.StartDirection : double | Start Direction |

## LandXML

### File format issues

Fixed in LandXML 1.2 schema line 1061. From <xs:complexType name="PointType" mixed="true"> to <xs:complexType name="PointType" mixed="false">. Otherwise this error occurs:



For more details see this website:

<http://www.w3schools.com/schema/schema_complex_mixed.asp>

A mixed complex type element can contain attributes, elements, and text.

**Complex Types with Mixed Content**

An XML element, "letter", that contains both text and other elements:

<letter>  
  Dear Mr.<name>John Smith</name>.  
  Your order <orderid>1032</orderid>  
  will be shipped on <shipdate>2001-07-13</shipdate>.  
</letter>

The following schema declares the "letter" element:

<xs:element name="letter">  
  <xs:complexType mixed="true">  
    <xs:sequence>  
      <xs:element name="name" type="xs:string"/>  
      <xs:element name="orderid" type="xs:positiveInteger"/>  
      <xs:element name="shipdate" type="xs:date"/>  
    </xs:sequence>  
  </xs:complexType>  
</xs:element>

**Note:** To enable character data to appear between the child-elements of "letter", the mixed attribute must be set to "true". The <xs:sequence> tag means that the elements defined (name, orderid and shipdate) must appear in that order inside a "letter" element.

We could also give the complexType element a name, and let the "letter" element have a type attribute that refers to the name of the complexType (if you use this method, several elements can refer to the same complex type):

<xs:element name="letter" type="lettertype"/>  
  
<xs:complexType name="lettertype" mixed="true">  
  <xs:sequence>  
    <xs:element name="name" type="xs:string"/>  
    <xs:element name="orderid" type="xs:positiveInteger"/>  
    <xs:element name="shipdate" type="xs:date"/>  
  </xs:sequence>  
</xs:complexType>

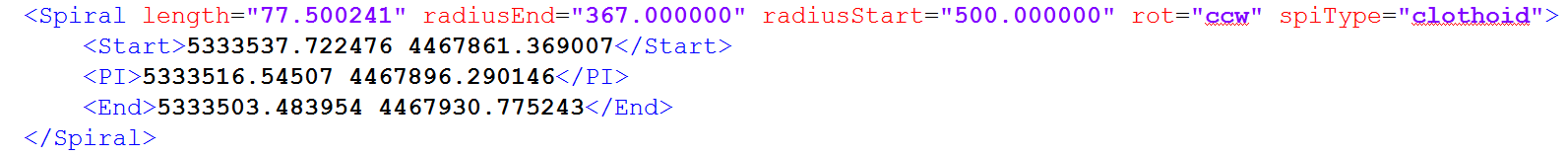
Spelling mistake: Weiner Bogen -> wiener bogen

### Clothoid



### Clothoid implementation

For simple cases e.g. clothoid begins or ends with curvature=0 the implementation is very easy. But implementing an arbitrary clothoid takes a bit more effort. Therefore this short tutorial helps to understand the magic behind the implementation. For the TUM Open Infra Platform the clothoid class has to deliver a getPosition(double lerpParameter) method. The instances are constructed out of the LandXML spiral element parameters.

Input parameters from LandXML:

start, end, pi (point of intersection = intersection of start- and end tangents), length, radiusStart, radiusEnd, clockwise/counterclockwise (cw/ccw)

Some general formulas for clothoids ():

radiusStart:

radiusEnd:

startCurvature:

endCurvature:

Calculate clothoid constant from LandXML input parameters:

Calculate lengths at and :

clothoid length:

length at :

length at :

How to compute local x, y coordinates at a given Length with Fresnel integrals:

How to compute global x, y coordinates at a given interpolation parameter

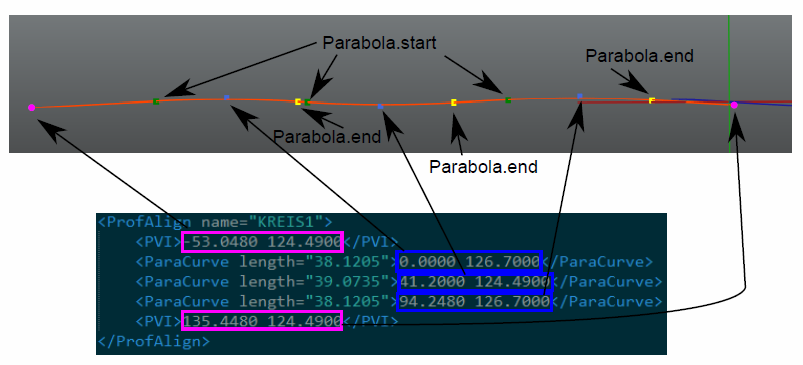
1. Calculate length at given interpolation parameter by interpolating between L1 and L2
2. Calculate offset x, y, coordinates at L1
3. Calculate deltaX and deltaY by calculating x, y coordinates at interpolation length(1.) and subtract the offset values
4. Calculate rotation angle between local and global coordinate system and rotate coordinates by this angle
5. Translate local system to global system by vector startgobal-startlocal



Future work: Convert in the minimum parameter Schema of RoadXML.

### Vertical alignment

A vertical alignment is described in LandXML by ProfAlign. LandXML does not store the start (green points) and end point (yellow points) of a parabola element. Instead it stores a so called PVI (point of vertical intersection, blue color). Besides the PVI also a length value is stored for each parabola. From the length value and the PVI the start and end point can be computed.



### Computation of vertical alignments (parabolas)

; ; ,

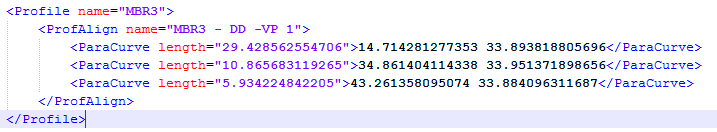
;

I nach b:

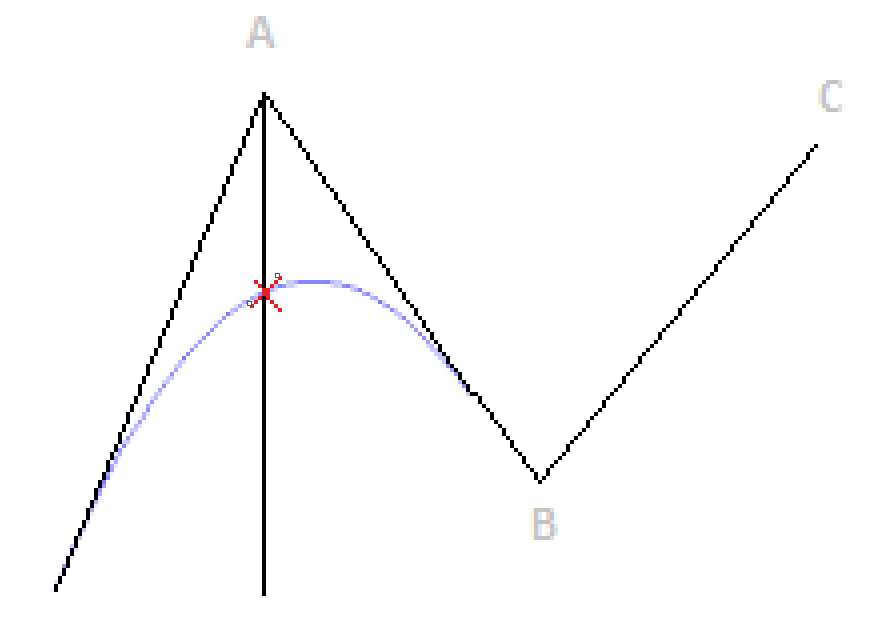
I in II:

a=-0,0075

### Vertical alignment that starts with a parabola

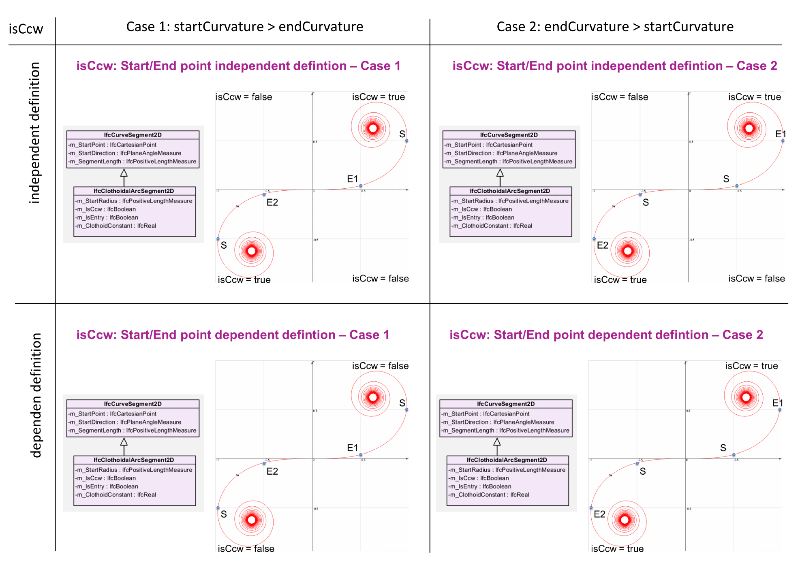


This is an invalid vertical alignment. Depending on the first tangent the red intersection point can be placed on different locations.



## IFC Alignment

Hier gilt die „dependent definition“ (untere Reihe) – das heißt das Attribut „IsCcw“ der Klothoide ist “start/end point dependent” und nicht “start/end point independent”



## RoadXML

RoadXML is an open file format for the logical description of road networks (<http://www.road-xml.org/>). It is used most frequently in traffic and driving simulations. RoadXML provides a data structure for tracks. A Track consist of a XYCurve and a SZCurve. The XYCurve describes the horizontal alignment of track and the SZCurve the vertical alignment.

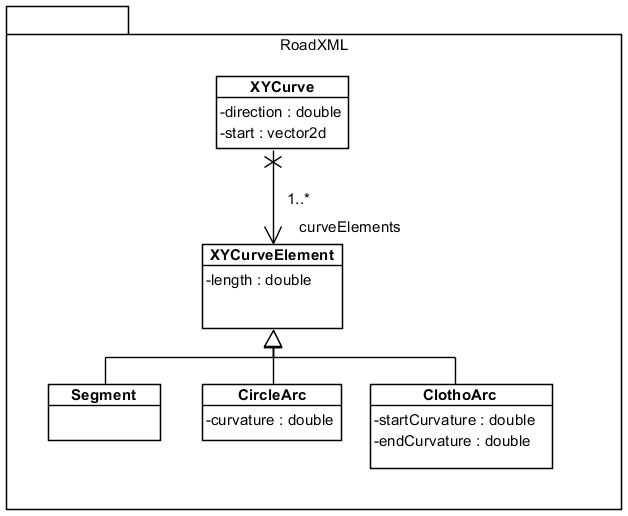


Figure 1: A XYCurve in RoadXML consist of staight lines (segment), circle arc and clothids (ClothoArcs).

<Track endNode="Intersection" name="Track 3" startNode="Intersection 2">

<XYCurve direction="1.75917" x="50.0104" y="-83.2074">

<ClothoArc endCurvature="0" length="8.009" startCurvature="-0.0309793"/>

<ClothoArc endCurvature="0" length="3.51943" startCurvature="0.0143293"/>

<CircleArc curvature="0.00105004" length="128.811"/>

<Segment length="2.7"/>

</XYCurve>

The direction specified in the <XYCurve> tag is the initial direction of the curve (in radians, 0 is pointing along the x axis: east, positive counter clockwise). The x and y values are the initial position of the curve, in meters.

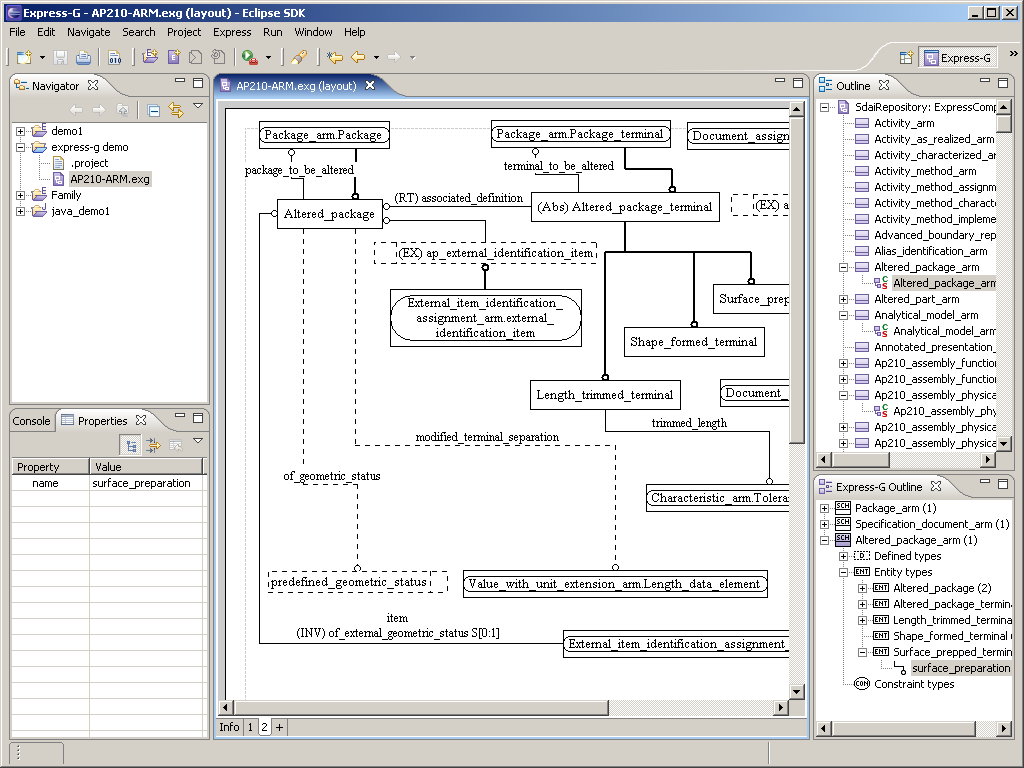
The circle arc starts in the direction and position specified in the <XYCurve> tag and has the specified curvature and length. Note that if you add another <CircleArc> tag after the first one, the initial position and direction of the following one will be the final positon/direction of the previous one [Mail from Hello Hewat].

## Tools

### JSDAI

For editing EXPRESS Schemas the tool JSDAI for Eclipse (<http://www.jsdai.net/eclipse>) can be used.

It provides a graphical editor for viewing EXPRESS diagrams (EXPRESS-G):



The tool can also validate the syntax of an express schema.

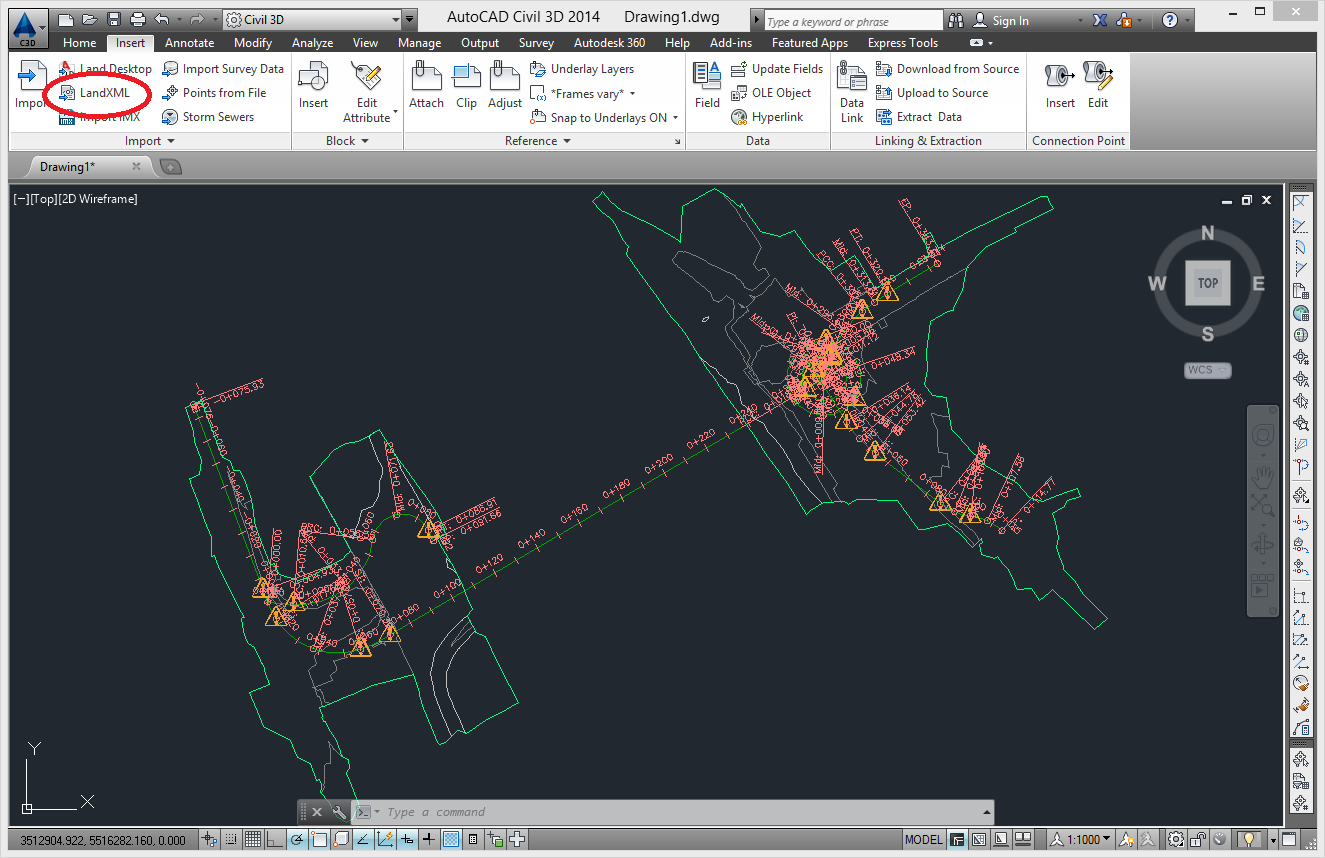
Moreover JSDAI provides an SDAI API for Java that enables Java developers to read and write step files for a given express schema.

### AutoCAD Civil 3D 2014

AutoCAD Civil 3D 2014 is a BIM tool for civil engineering (<http://www.autodesk.com/products/autodesk-autocad-civil-3d/features/all/gallery-view>)

Students can acquire a free license from <http://www.autodesk.com/education/student-software>. Just setup an account. Afterwards you have free access to all Autodesk products (including 3ds max, Maya, AutoCAD Civil 3D 2014 and many more).

AutoCAD Civil3D 2014 can import LandXML files.



# Code Guidelines

## Syntax #1

Use nullptr instead of NULL.

## Ident style #1

Use Allman style.

int OpenInfraPlatform::DataManagement::IfcZipper::writeLocalFileHeader(localzipheader\* header) {

if (!ofs.is\_open()) {

BLUE\_LOG\_STREAM\_EX("IfcZip Export", buw::eLogSeverityLevel::Error) << "Output filestream is closed";

return -1;

}

if (header == NULL) {

BLUE\_LOG\_STREAM\_EX("IfcZip Export", buw::eLogSeverityLevel::Error) << "header is null";

return -2;

}

ofs.write((char\*)header, 14);

ofs.write((char\*) &header->crc32, 16);

int pos = filename.toStdString().find\_last\_of('/', filename.size()) + 1;

ofs.write(filename.toStdString().substr(pos,header->fileNameLength).data(), header->fileNameLength);

return 0;

}

Should be:

int OpenInfraPlatform::DataManagement::IfcZipper::writeLocalFileHeader(localzipheader\* header)

{

if (!ofs.is\_open())

{

BLUE\_LOG\_STREAM\_EX("IfcZip Export", buw::eLogSeverityLevel::Error) << "Output filestream is closed";

return -1;

}

if (header == NULL)

{

BLUE\_LOG\_STREAM\_EX("IfcZip Export", buw::eLogSeverityLevel::Error) << "header is null";

return -2;

}

ofs.write((char\*)header, 14);

ofs.write((char\*) &header->crc32, 16);

int pos = filename.toStdString().find\_last\_of('/', filename.size()) + 1;

ofs.write(filename.toStdString().substr(pos,header->fileNameLength).data(), header->fileNameLength);

return 0;

}

## Comments #1

Write comments in English language – do not use another language!

// Schreibt den übergebenen File Header in die Datei

should be:

// Stores the given file header in the file

## Comments #2

Use a space to spate the comment introduction in single-line comment form the real comment.

//Stores the given file header in the file

Should be:

// Stores the given file header in the file

## Tags

Each source file should have the following header:

/\*! \verbatim

\* \copyright Copyright (c) 2015 Technische Universität München

\* Chair of Computational Modeling and Simulation. All rights reserved.

\* \author Fabian Schöttl <fabian.schoettl@tum.de> (https://www.cms.bgu.tum.de/en/)

\* \author Julian Amann <julian.amann@tum.de> (https://www.cms.bgu.tum.de/en/team/amann)

\* \author Dominic Singer <dominic.singer@tum.de> (https://www.cms.bgu.tum.de/de/team/singer)

\* \brief This file is part of the TUM Open Infra Platform.

\* \endverbatim

\*/

See also “OpenInfraPlatform\docs\tag.txt” file. Use the same formatting as is used in the “tag.txt” file.

Add your name to the author list if you have contributed to the corresponding file. Try to sort the author list by contribution. If you have made the main contribution to a file your name should be listed first.

## Includes #1

Use „/“ instead of „\“.

#include "BlueFramework/Infrastructure/Alignment/Alignment2DBased3D.h" // correct!

#include "BlueFramework\Infrastructure\AlignmentModel\AlignmentModel.h" // wrong!

## Includes #2

Do not use .. for a relative path in an include file. When moving source files this path names get invalid. Forthermore it is difficualt to figure out where a file is exactly located and how it is related to the other include files.

#include "../Benchmark.h" // wrong!

#include "OpenInfraPlatform/Benchmark.h" // correct!

## CMake

Use lower case characters, because they are easier to type and modify. So instead of

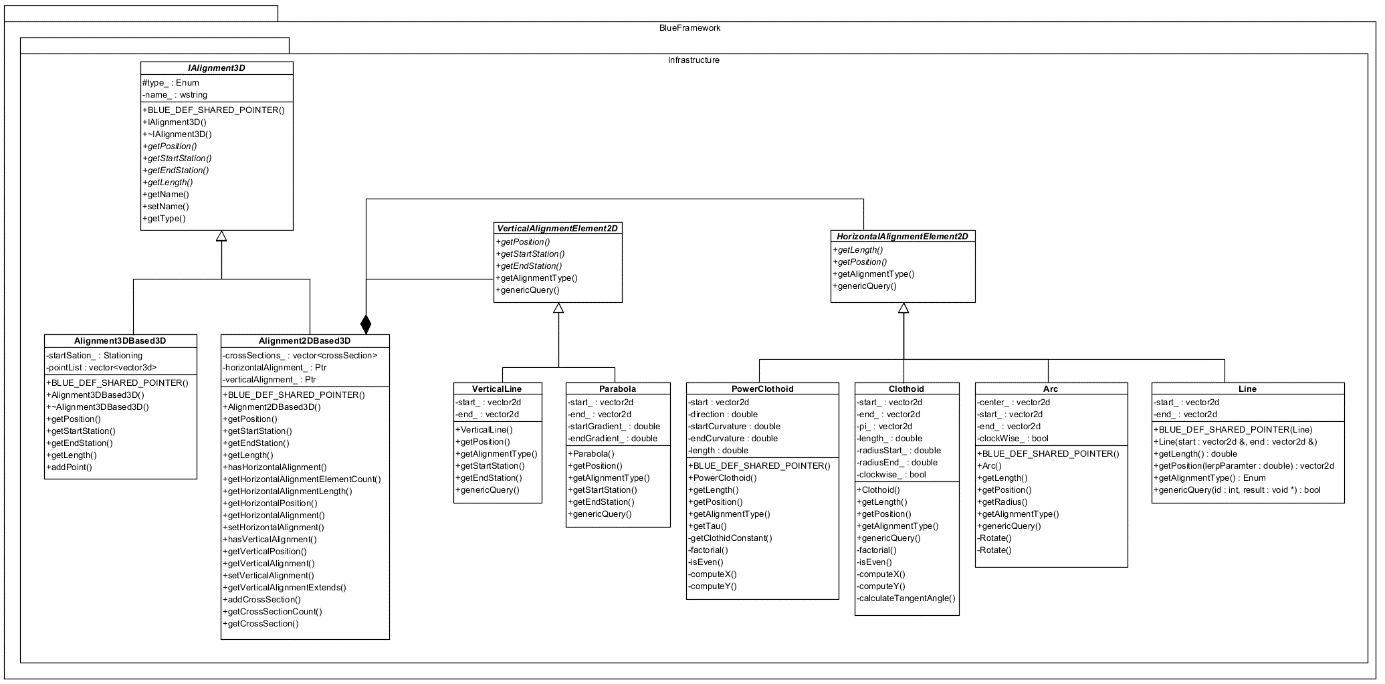
PROJECT(BlueFramework.Infrastructure)

Use:

project(BlueFramework.Infrastructure)

Also mixed lower and upper case looks strange. Stick to one.

# Code Architecture



There are two types of alignments – a 3D based one – and a 2D based one. A 2D based one consists of a vertical alignment and a horizontal alignment. A horizontal alignment consist of horizontal alignment elements such as PowerClothoid, Arc or Line. A line consist of a start and an end point.

# Code in Detail

## IFC Eearly Binding Meta Template Library (ifcEMT)

wir sollten eine Art IFC Eearly Binding Meta Template Library (ifcEMT) entwickeln

Für jeden Entity Typ wird ein Template Parameter generiert – etwa so:

namespace BlueFramework

{

       namespace Ifc2x3

       {

             class IfcFace;

             ...

       }

}

namespace emt

{

       template <

             typename IfcFaceTypeType

             ...

       >

       struct IfcEntityTypes

       {

             typedef IfcFaceTypeType IfcFace;

             ...

       };

       typedef IfcEntityTypes<

BlueFramework::Ifc2x3::IfcFace

...

>

Ifc2x3EntityTypes;

       typedef IfcEntityTypes<

BlueFramework::Ifc4::IfcFace

...

>

Ifc4EntityTypes;

}

Am Schluss haben wir dann einfach ein Template mit ca. 600 Template Parametern - ist auch schon eingecheckt – siehe EMTIfcEntityTypes.h, EMTIfc2x3EntityTypes.h, EMTIfc4EntityTypes.

Ich habe begonnen aus dem FaceConverter ein Template zu machen:

             template <

                    class IfcEntityTypesT,

                    class IfcUnitConverterT

             >

             class FaceConverterT

Ähnlich bin ich beim PlacementConverter vorgegangen:

template <

                    class IfcEntityTypesT

             >

             class PlacementConverterT

             {

Zum Testen benutze ich den FaceConverterT in der Methode convertIfcSurface der Klasse FaceConverter.

                    void convertIfcSurface(const shared\_ptr<OpenInfraPlatform::Ifc2x3::IfcSurface>& surface,

                           const carve::math::Matrix& pos,

                           shared\_ptr<carve::input::PolylineSetData>& polyline\_data)

                    {

                           FaceConverterT<emt::Ifc2x3EntityTypes, OpenInfraPlatform::Ifc2x3::UnitConverter> f(

m\_geom\_settings, m\_unit\_converter, m\_curve\_converter);

                           f.convertIfcSurface(surface, pos, polyline\_data);

                           return;

Kannst du das Template FaceConverterT noch vollständig implementieren und testen?

Kannst du außerdem noch das PlacmentConvertT Template vollständig implementieren?

Im Prinzip muss dabei nur jedes vorkommen eine IFC Entity Typs ersetzt werden – etwa so:

IfcPlacement

->

typename IfcEntityTypesT::IfcPlacement

Soweit ich das jetzt überblicke müssen noch folgende Klassen in Templates umgewandelt werden (genau in dieser Reihenfolge):

CurveConverter

ProfileCache

ProfileConverter

RepresentationConverter

SectionedSpine

SolidModelConverter

StylesConverter

IfcImporter

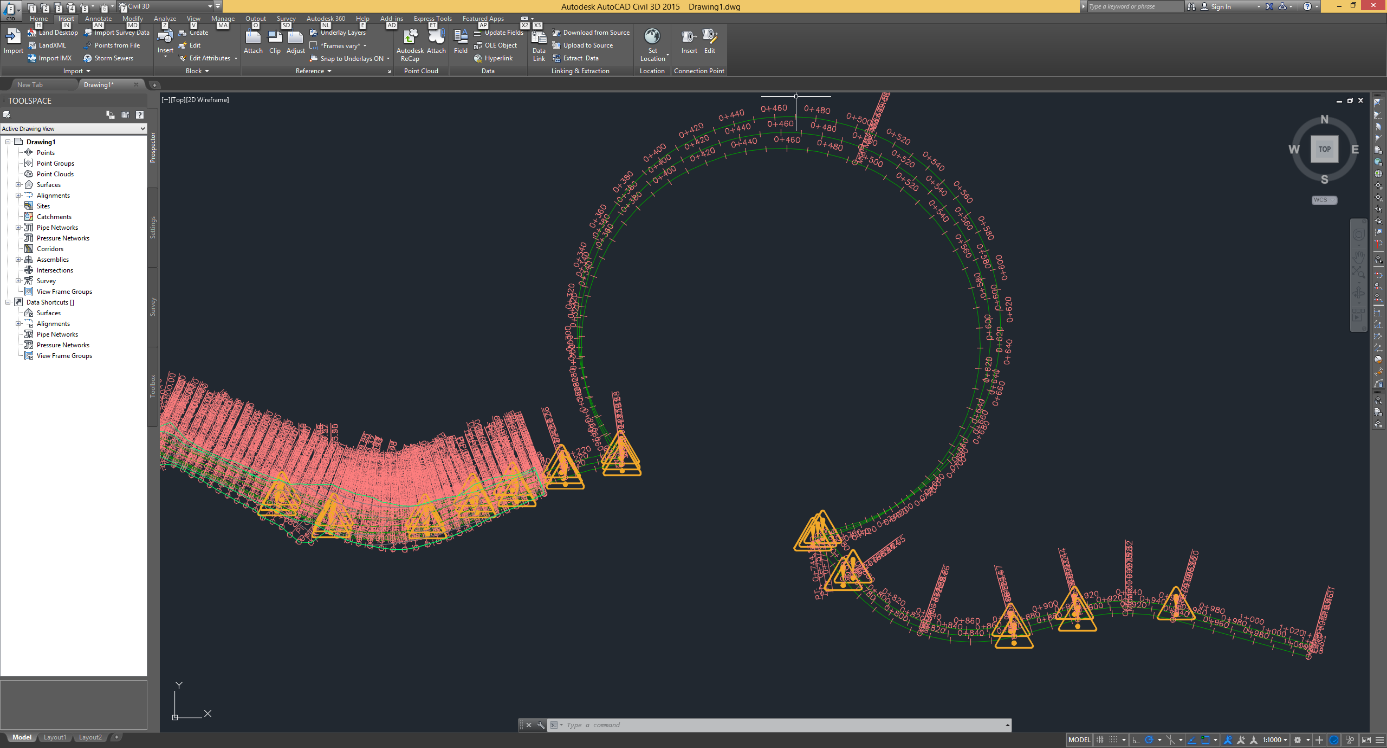
BlueFramework.Viewport anbieten

# Testing

## Testdata

testdata\CityCad\Sito\_Tie.xml

Autodesk Civil 3D cannot handle this file:



When trying to improting this file to OIP we detected that we are not able to compute the end point of the following clothoid:

<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>

<LandXML xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.inframodel.fi/inframodel http://www.inframodel.fi/schemas/2.0.9/inframodel.xsd" xmlns="http://www.inframodel.fi/inframodel" date="2013-06-05" time="16:32:35" version="1.2" language="Finnish" readOnly="false">

<Project name="InfraModel" desc="InfraModel testiaineisto" state="proposed"/>

<Units>

<Metric areaUnit="squareMeter" linearUnit="meter" volumeUnit="cubicMeter" temperatureUnit="celsius" pressureUnit="mmHG" directionUnit="grads" angularUnit="grads" diameterUnit="meter" widthUnit="meter" heightUnit="meter" velocityUnit="kilometersPerHour" />

</Units>

<CoordinateSystem horizontalCoordinateSystemName="KKJ" verticalCoordinateSystemName="N60" desc="National Land Survey Finland " rotationAngle="0" />

<Application name="CityCad" manufacturer="Sito" manufacturerURL="http://www.sito.fi" version="5" timeStamp="2013-06-05T16:32:35">

<Author createdBy="matti.heikkila" timeStamp="2013-06-05T16:32:35" company="Sito" companyURL="http://www.sito.fi" />

</Application>

<Alignments name="MT301" desc="Inframodel3">

<Alignment name="Mittalinja" oID="5363" staStart="0" length="25.6000000000951" state="proposed">

<CoordGeom>

<Spiral length="25.6000000000951" radiusEnd="INF" radiusStart="90" rot="ccw" spiType="clothoid" constant="48.0000000000891" dirEnd="76.0074731892313" dirStart="85.0616210292649" staStart="141.788351337172">

<Start>6677640.57582574 2546177.99634336 0</Start>

<PI>6677644.56796775 2546190.1400906 0.0</PI>

<End>6677648.8511542 2546202.19761461 0</End>

</Spiral>

</CoordGeom>

<Profile staStart="0">

<ProfAlign name="0">

<PVI>0 1</PVI>

<PVI>25.6000000000951 1</PVI>

</ProfAlign>

</Profile>

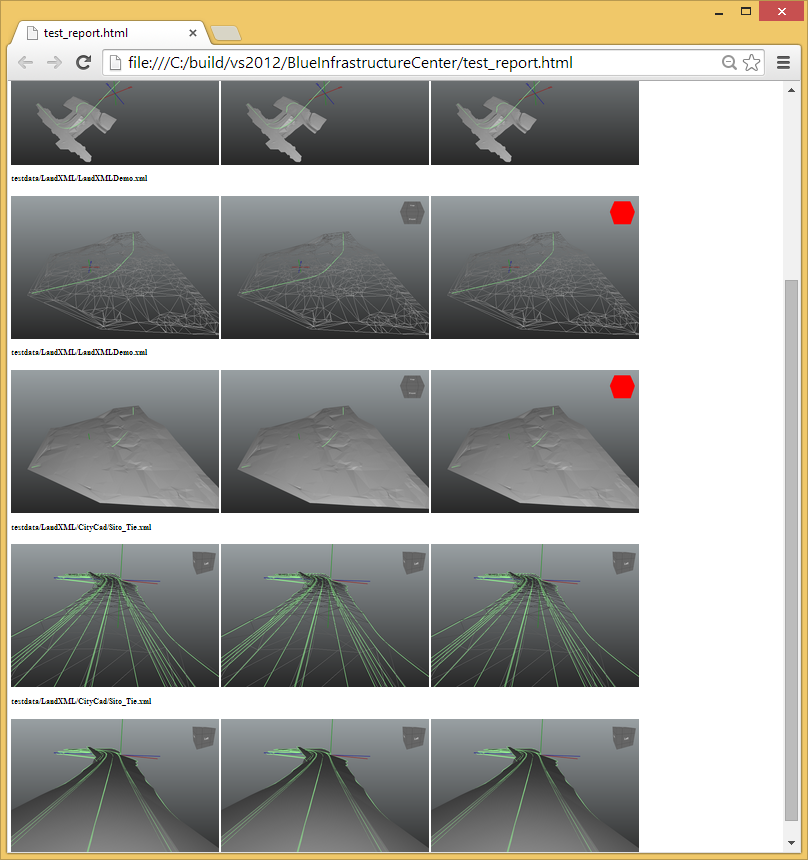
</Alignment>

</Alignments>

</LandXML>

It seems the provided data is invalid. Maybe this is because of an export error in CityCad.

## Unit Testing



## IFC Alignment Common Mistakes

### Check #1

Fehlende GUIDs (z.B. in vertical\_alignment.ifc)

#23=IFCALIGNMENT($,$,$,'test',$,$,$,$,#24,#34,$);

GUID ist kein optionales Attribut und sollte daher gesetzt werden

### Check #2

RelatedElements mindestens SET [1:?] (z.B. vertical\_alignment.ifc)

#20=IFCRELCONTAINEDINSPATIALSTRUCTURE('09AjNQa69Dlv5Mt8ZRJ1R3',$,$,$,$,#16);

### Check #3

NaN Werte (z.B. terrain-and-alignment.ifc)

#102=IFCALIGNMENT2DVERSEGLINE($,$,$,265.65600000000001,0,125.815,-1.#IND);

### Check #4

INTEGER anstatt DOUBLE – einige EXPRESS-Parser sind da ganz genau (z.B. vertical\_alignment.ifc)

#29=IFCCLOTHOIDALARCSEGMENT2D(#30,5.7380475130104633,77.500241000000003,500,.T.,.T.,326.99697894146078);