OST
Ostschweizer
Fachhochschule

Prototype of an open GIS-to-BIM Converter

Semester Assignment

Department of Computer Science
OST – University of Applied Sciences
Campus Rapperswil-Jona

Spring Term 2023

Author(s)	Jamie Maier
Advisor	Prof. Stefan F. Keller
Project Partner	Reto Senn (Bitforge AG)
	Prof. Martin Beth (OST)

ABSTRACT

Underground utility networks are commonly managed in 2D Geo-Information Systems (GIS) using normed formats like the SIA 405-INTERLIS standard in Switzerland. However, there is a growing demand for representing these utility networks in a 3D format, particularly in the context of Building Information Modeling (BIM). This research presents a Python Command Line Interface (CLI) prototype that enables the conversion of 2D INTERLIS data to valid 3D data in the Industry Foundation Classes (IFC) format.

The resulting prototype has a CLI that allows users to input a GeoPackage file, specify a coordinate range, and define the output path for the resulting IFC file. It validates the provided arguments and performs the conversion process using Python libraries such as GeoPandas and IfcOpenShell. The resulting 3D data includes additional properties for visualization, such as underground depth, element diameter, and color schemes defined as Intelligent Defaults.

The prototype lays a solid foundation for future development. The outlook includes further enhancements, such as marking intelligent default data for clarity of inaccuracies, enhancing the Intelligent Defaults with additional characteristics, supporting other utility network mediums, and adding configurability and extensibility features for users. Overall, this research demonstrates the feasibility of converting 2D INTERLIS data to a 3D IFC format, opening up possibilities for improved representation and visualization of underground utility networks in the field of BIM.

Keywords: 2D to 3D conversion, INTERLIS, IFC, Python CLI, GeoPackage, Building Information Modeling (BIM).

Executive Summary

Starting Situation

Underground utility networks are usually managed in Geo-Information-Systems (GIS) in a 2-dimensional format. This data is described in Switzerland in a normed form, described by the SIA 405-INTERLIS standard [1]. The SIA 405-INTERLIS standard contains multiple models for different utility pipe mediums such as sewage, electricity, gas etc. which contain detailed information for the utility elements in these categories. Besides these models, there exists a minimalistic data exchange approach model LKMAP, that provides the basic locational and dimensional properties of pipe cadaster elements.

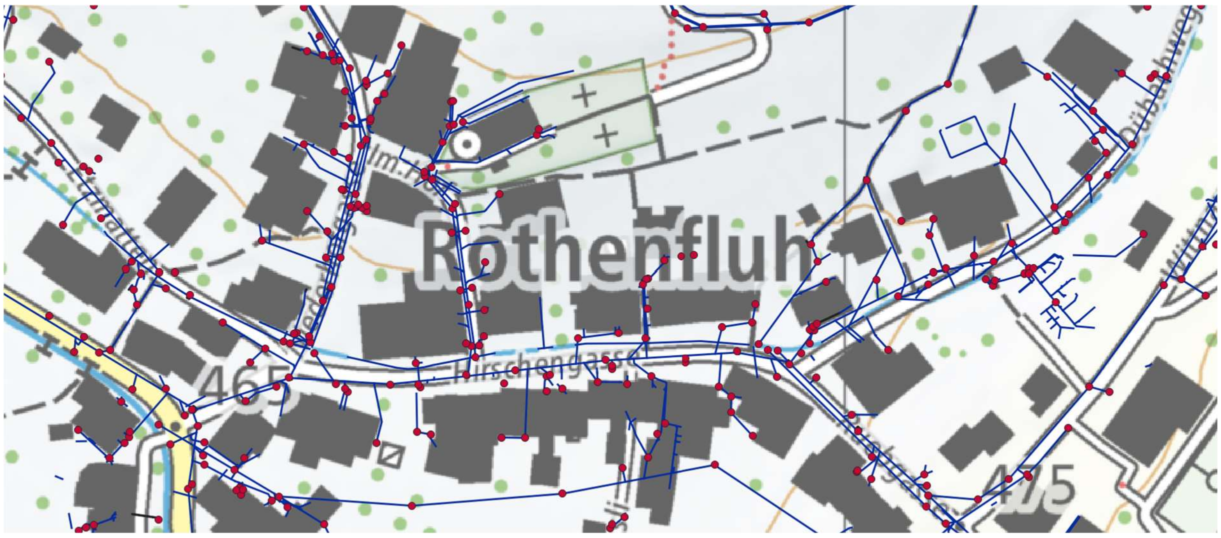

Figure 1: An excerpt of 2D data in Rothenfluh BL. Sewage ducts displayed as LKPunkt (red) and pipes displayed as LKLinie (blue)

However, there is a high demand for these pipe cadasters to be available in a 3 dimensional format. This for example would be a huge benefit in Building Information Modeling, a process for generating and managing digital representations of physical and functional properties of building sites in 3D. The pipe cadaster data would allow BIM operators to plan, model and design the exact locations where utility pipes intersect the building of their model.

Following this need for 3 dimensional representations of pipe cadasters this thesis aims to provide a Command Line Interface (CLI) Prototype that lets users convert 2D INTERLIS data to valid 3D data in the IFC format.

Approach

The initial steps taken for this project were conducting research into the standards INTERLIS and IFC, finding libraries that support the different format/data conversion steps and acquiring test data to be worked with.

With this knowledge build up, it was decided to split the project into four different solution strategies: the command line interface, importing given data, adding missing values to given data to support 3D visualization and the actual writing of the output format IFC with the data.

The first solution concept aimed to create a CLI that requires the user to input a LKMAP data file (.xtf) and the corresponding model file(.ili). With a required coordinate range argument to minimize the amount of data. The core idea for the CLI was to keep it as simple as possible to ensure easy usability for users.

The concept defined to import data was to convert INTERLIS transfer data into a GeoPackage using the Ili2GPKG library [2] in Python and then accessing this data with the GeoPandas [3] library. To be able to use Ili2GPKG in Python the idea was to convert the library into a native-image with GraalVM [4], which was eventually deemed out of scope due to time constraints and errors occurring.

Because the import data is in a 2D format it was necessary to enhance it for proper 3D visualization. This lead to the concept of "Intelligent Defaults", which added properties of depth underground, diameter of elements and color schemes for different waste water types to the existing data.

The final step was the concept of creating IFC 4.3 files. This was to be achieved with the Python library IfcOpenShell [5]. This was the largest part of the project, due to the extensive possibilities provided by IFC. One of the discussions was which element type in IFC would be used to create the LKObjects. The options consisted of three different approaches, a concrete approach mapping the element types one to one, an abstract approach that contains the concrete realm the objects belong to, but not a one to one mapping of the object types and a fully abstract approach working with proxies that can be anything anywhere. Because LKMAP is also an abstract approach that defines objects in a specific realm, it was decided to use the second option mentioned above.

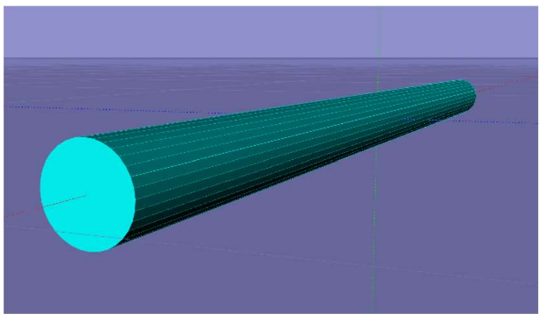
Figure 2: Initial Visualization of Pipe using IfcOpenShell (Screenshot made in LexoCAD)

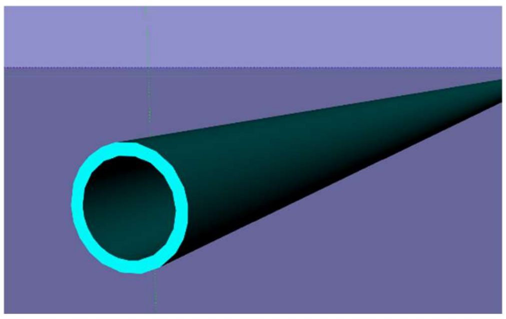
Figure 3: Final Visualization of Pipe using IfcOpenShell (Screenshot made in LexoCAD)

Considerations were also made about the visualization of the IFC elements. The figure above on the left shows the first attempts to create a pipe representation in IFC, while the figure above on the right shows the final representation chosen for pipes in IFC. It works by taking two coordinates pulling a line between them (just like in the original 2D data) and then extruding a disc of a given diameter around this line.

Result

The resulting CLI allows the user to input a GeoPackage, a coordinate range for the objects and the path to where the IFC file should be generated. It validates that all the arguments are given, that the GeoPackage can be found and that the IFC file does not already exist. Executing the command convert-gpkg-to-ifc with the above-mentioned arguments the result is a valid IFC 4.3 file that can viewed in a BIM viewer.

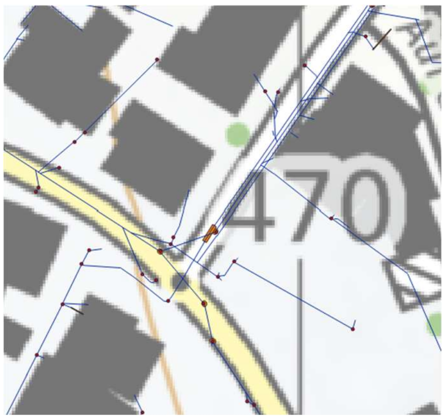

Figure 4: Original 2D view of the wastewater utility elements of Anwilerstr. 10, BL

Because the concept of converting ILI2GPKG to a native image with GraalVM was let go the CLI requires a GeoPackage instead of the originally planned LKMAP data.

The outcome of importing data therefore included a FilterChain consisting of filters applied to the given GeoPackage. The first of these filters minimizes the number of data objects with the coordinate range CLI argument, the next filter groups these into their respective LKObjekt types and the final filter adds characteristic attributes to the objects.

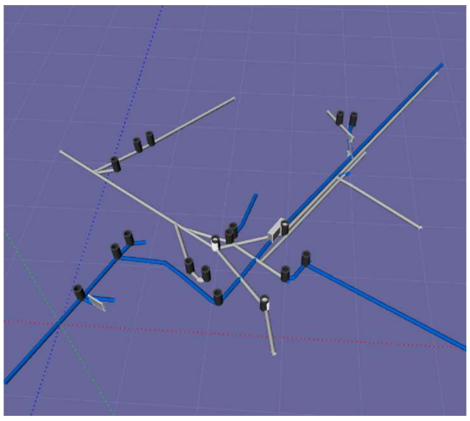
Figure 5: Wastewater elements exported as IFC 4.3 file by the prototype.

Further, Intelligent Defaults enhance the provided data with locational heights, widths of elements, color schemes and scaling to set the origin point of the coordinate system to the minimum values of the provided range.

Finally, IFC element builders using IfcOpenShell create the elements from the processed data and allocate these to a created IFC 4.3 file.

Outlook

The project in its current form provides a good foundational prototype for further development. Facing the fact that the goals of using a native-image of the ILI2GPKG library was not fulfilled in this project, the main outlook is to complete this goals by fully proving the defined POCs and integrating these concepts into the existing prototype, which will happen as part of the follow-up bachelor thesis.

With this foundation set, there are various improvements to be implemented. The usability of the prototypes generated IFC file can be improved by marking default data, in order to make certain that users are not led to believe that all the data is accurate. Intelligent Defaults can be enhanced with flow direction, altitude gradients and further characteristics for more probable results. Another enhancement would be to have functionality to support the creation of IFC files with the other LKMAP utility network mediums.

Further possibilities include the functionality to provide the user with configurability for intelligent defaults or the functionality for users to add additional GeoPackage data to existing IFC files.

Acknowledgements

I would like to thank the following people for aiding with this semester thesis:

Prof. Stefan F. Keller for guidance and supervision during the process of this semester assignement.

Reto Senn for sharing his vision of the end result for the project and his experience with IFC.

Prof. Martin Beth for bringing domain specific knowledge and his experiences when working with IFC.

Prof. Lukas Schildknecht for his valuable insights on IFC discussions.

Pascal Knecht and Daniel Maier for proofreading this semester assignment.

Contents

PART I

1 Introduction

Today, underground utility networks like sewer pipes are managed in Geo-Information-Systems (GIS) in a normed Form. In Switzerland this normed form is described by the 2D SIA 405-INTERLIS standard [1]. However, in construction CAD systems 3D-data is required. This data should be in the IFC-format used in Building Information Modeling (BIM).

In the current situation these pipe cadasters are not fully compatible with BIM due to them not being 3D or in the appropriate IFC format. The 2D data format can be used as a baseline for planning with BIM however the result leaves much to be desired. The pipe cadasters are displayed as highly abstracted lines with attributes such as the diameter of the pipe, leading to inaccuracies and incomplete models.

Another problem arises in this context concerning data exchange formats and not just the dimensions of the data. The Swiss data exchange format INTERLIS is not supported in the BIM world. The compatible format used in todays BIM standards is IFC. To work around this issue, the tedious task of converting 2D to 3D data from one format to another is done manually by using the insufficient attributes such as target depth and shaft depth to create usable 3D data with assumed values.

This creates the issue of the data not being accurate and/or reliable. Due to the lack of information provided from the GIS 2D side many assumptions must be made about the 3D data, leading to additional inaccuracies and inconsistencies depending on the selected assumptions. Furthermore, using this approach gives the false impression that the data that was assumed, such as the target depth in the ground, diameter, connection points, etc. is accurate and reliable.

1.1 Assignment

This thesis has the primary objective of creating a prototype of an open LKMAP-to-IFC Converter with a Command Line Interface (CLI).

ASSIGNMENT

GOALS	The main goal of this project is to gain experience with GIS-BIM-Integration. Particularly with the Swiss pipe cadaster norms converted into 3D IFC formats, with the simplicity of a command line interface. Further, the intention is to evaluate existing technologies and libraries that support the desired functionality fully or partially. As a note this thesis has the general-purpose of acquiring the skills needed in everyday software projects, like working with agile project management, versioning, testing, CI/CDs and documenting.

CHALLENGES	The core challenges in this thesis are how to map LKMAP data to IFC data and how this data needs to be enhanced to successfully display 3D visualizations. Part of these core challenges are the handling of missing data (like altitude, not provided in 2D formats), displaying unreliability of assumed or enhanced data and the possibility of deriving further properties from 2D to enhance the 3D visualization, for example the materials of the 2D elements.
CONDITIONS	The thesis should use LKMAPs (Sewage Section) newest model version that has valid test data available. The programming language to be used should be Python3 , with PEP8 styling. The resulting application should be tested on Windows and optionally on a Linux operating system. The developed software should be open-source software, preferably with a permissive license.

Table 1: Assignment

1.2 Basic Conditions

This thesis was done as a semester assignment (Studienarbeit). A time budget of 240 hours is reserved for the work on this assignment and will be rewarded with 8 ECTS credits.

2 Research & Evaluation

This chapter describes the process of getting to know the problem domain of this thesis and the evaluation of state of the art standards and technologies that already exist. The Research begins with the standards that are defined to be compatible with the application. Furthermore, existing technologies for working with these standards are examined, including available visualization Tools.

2.1 Standards & Formats

The standards & formats described in this chapter were part of the requirements for this project and therefore hold a large relevance to the final result of this project.

2.1.1 INTERLIS

The standard used for the data to be converted is INTERLIS. INTERLIS is a Swiss model-based data description language and exchange format. It consists of an .ili File that describes the schema of the data and an exchange file of the type .itf. It is used for modelling spatial data accurately. INTERLIS has been anchored in the Swiss Geoinformation Legislation since 2008. INTERLIS most current version is 2.3, which contains the object-oriented approach for data models. [6]

INTERLIS has the asset of using an object-oriented approach making it easier to understand the relationships between objects and their attributes. However, it should be noted that there is only a small number of tools and libraries that support this standard, because it is Switzerland specific.

2.1.2 LKMAP SIA 405

The SIA 405 norm, a norm defined by the Swiss Engineer and Architect Association (SIA), is meant for the exchange and publication of pipe cadaster data. It defines the minimal requirements for processing this data in a GIS application. [1]

For this thesis the focus is on LKMAP, which describes the minimal information needed for cadastral registers. The scope of these LKMAPs in this project was defined down to pipe cadasters (in particular sewage pipes), which include information about the location of these pipes and other properties relevant for minimal visualization. They are described by LKObjekte which can either be a LKFlaeche (Area), LKPunkt (Point) or a LKLinie (Line). Furthermore, these objects can contain describing attributes like characteristics, accuracy of location, etc.

LKMAP provides a good abstraction level for cadastral registers which was a large help for this prototype. This because the focus was on geometry and not the specifics of the objects provided.

2.1.3 GeoPackage

GeoPackages are a geospatial data storage format that combines the advantages of vector and raster data within a single file. They are based on the SQLite database format, allowing for efficient storage, indexing, and retrieval of geospatial information. GeoPackages support the storage of multiple layers, including points, lines, polygons, and raster datasets, making them versatile for a wide range of geospatial applications. These files are platform-independent and can be easily shared, accessed, and manipulated across different software and devices. The OGC GeoPackage specification provides a comprehensive definition of the format, outlining its structure, data types, and supported operations, ensuring interoperability and compatibility across various geospatial tools and platforms [7]

The usage of the GeoPackage data storage format in the project was extensive. The data was easily accessible with the help of GeoPandas (see Chapter 2.2.5). The only detriment found while using GeoPackages was the fact that geometry is stored in a BLOB (Binary Large Object) data type making it unreadable directly in the database.

2.1.4 IFC

IFC (Industry Foundation Classes) is a file format used in BIM. The result of the application this project aims to achieve is such an IFC file. This is due to the wide usage and integration of these IFC files in BIM capable software. Another aspect of IFC that further supported its use in this project was that IFC is capable of modelling in 3D, which INTERLIS does not support.

To begin working with IFC an IFC schema to be used needs to be defined. This schema contains a collection of definitions and rules that define the structure and the behavior of the data contained within the file. Furthermore, the schema describes the data model and the classes of objects that can be represented, including relationships between these objects. Additionally, properties and attributes and their representations are defined in such a schema. The main purpose of an IFC schema is to ensure data consistency and interoperability between different software tools, making collaborations on projects effective.

IFC4.3

IFC4.3 is the latest version of IFC supporting a broad spectrum of BIM data. Compared to other IFC schemas IFC4.3 is not defined for a specific domain in the BIM universe. Some of the key features in the IFC4.3 schema that are relevant for this thesis are enhanced-geometry support and expanded coverage of building systems. The enhanced geometry support includes curved elements and parametric shapes. And the expanded coverage of building systems provided new object classes supporting electrical, mechanical and pluming components. [8]

IFC4.3 was a predefined requirement of the assignment, due to LKMAP SIA 405 (Chapter 2.1.2) being a model with minimal data needed to successfully describe pipe cadasters, the decision makes sense that an IFC schema would be chosen that supports the minimal valid data needed to successfully represent the objects of LKMAP.

2.2 Technologies & Libraries

This chapter focuses on the technologies and libraries that were considered for the solution concept of the prototype. Some of them were considered but eventually not used in the result of this project.

2.2.1 INTERLIS to DB

The first data conversion step that is needed for a 2D to 3D mapper is to get the 2D data, which is given in a specific Format/Standard (in the case of this thesis INTERLIS), into a manageable state. Since INTERLIS is a very specific standard the logical process is to examine existing Technologies & Libraries that support reading INTERLIS formats and converting them to some kind of database.

Several options were considered, such as Ili2pg (for PostgreSQL/PostGIS), ili2gpkg (for GeoPackage), and ili2fgdb (for ESRI FileGDB) are programs that write INTERLIS-transfer files according to an INTERLIS-model file into a database or create such transfer files from an existing database. [9] Other options such as ili2FME that include proprietary source code were also found.

ILI2PG

Since PostgreSQL was the most familiar option of the above mentioned 3, this option was examined first. However, due to the need of a running PostgreSQL server and JDBC driver that supports SCRAM authentication methods this option was not chosen.

ASSETS	• Easy to use in QGIS (see 2.4.1) with Model Baker Plugin • Has an extension PostGIS for spatial data • Relational Database
DETRIMENTS	• Requires a JDBC Driver version that supports SCRAM authentication • Requires you to have a running PostgreSQL server

Table 2: ILI2PG Assets & Detriments

ILI2GPKG

The next alternative that was examined was the library for converting INTERLIS to a GeoPackage. Given the requirement of using the programming language Python for this thesis which supports the usage of GeoPandas (see 2.2.5), which in turn supports GeoPackage manipulation, and the fact that all other requirements were met, this library was decided to be the most fitting for this thesis.

ASSETS	• Easy to use in QGIS (see 2.4.1) with Model Baker Plugin • Creates a SQLite embedded database
DETRIMENTS	• All coordinates are stored in one column as a BLOB

Table 3: ILI2GPKG Assets & Detriments

Honorable Mentions

- ili2fgdb was not considered after the decision that ili2gpkg would be sufficient.
- ili2fme ("The plugin ili2fme enables the software package FME of Safe Software Inc. to read, write and display INTERLIS data" [10]) was not examined on account of it being part of a non-open-source product.

2.2.2 GraalVM

In order to use the Ili2GPKG library in python the idea was constructed that a native-image could be created from the library. This functionality is available in GraalVM.

GraalVM's native-image functionality is a powerful feature that allows developers to compile their applications into standalone native executables. By leveraging ahead-of-time (AOT) compilation, native-image eliminates the need for a virtual machine or just-in-time (JIT) compilation during runtime, resulting in faster startup times and reduced memory consumption. It achieves this by analyzing the application's code and dependencies to create a self-contained executable that includes only the necessary components [4].

GraalVM was let go as an option for this prototype due to errors and the time constraint of finding a workaround for these. One of these errors , GR-43413 [11], is currently being worked on by the GraalVM community while the others could be solved by ensuring that the library to build as a native-image is first compiled by GraalVM specifically.

2.2.3 GDAL/ogr2ogr

After first attempts with ili2pg failed, due to detriments mentioned in the Chapter 2.2.1, the GDAL library supporting file translations was examined.

GDAL contains a Vector Program, ogr2ogr. "ogr2ogr can be used to convert simple features data between file formats. It can also perform various operations during the process, such as spatial or attribute selection, reducing the set of attributes, setting the output coordinate system, or even reprojecting the features during translation." [12]

With ogr2ogr it was possible to map data of a GeoPackage into a PostgreSQL DB. However, it was also decided that this library would not be used, since data processing is possible through GeoPackages directly.

2.2.4 UML/INTERLIS Editor

The UML/INTERLIS editor allows the creation of a UML diagram that is then converted and exported into an INTERLIS-model. It also allows for a UML diagram visualization of existing INTERLIS-models by importing those. The editor uses the INTERLIS compiler and therefore also allows for validations of the created INTERLIS-models. [13]

For initial know-how building concerning LKMAP this tool was very practical. The two images shown are generated class diagrams using the UML/INTERLIS Editor. The first of the two showing the minimalistic diagram and the second showing the same model just with attribute types, cardinalities and association names defined in the model.

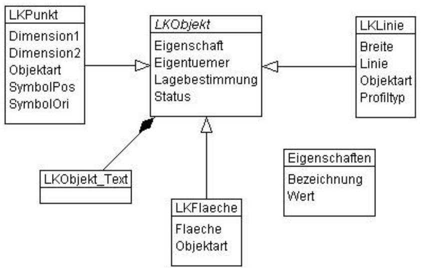

Figure 6: UML Class Diagram Screenshot from UML/INTERLIS Editor

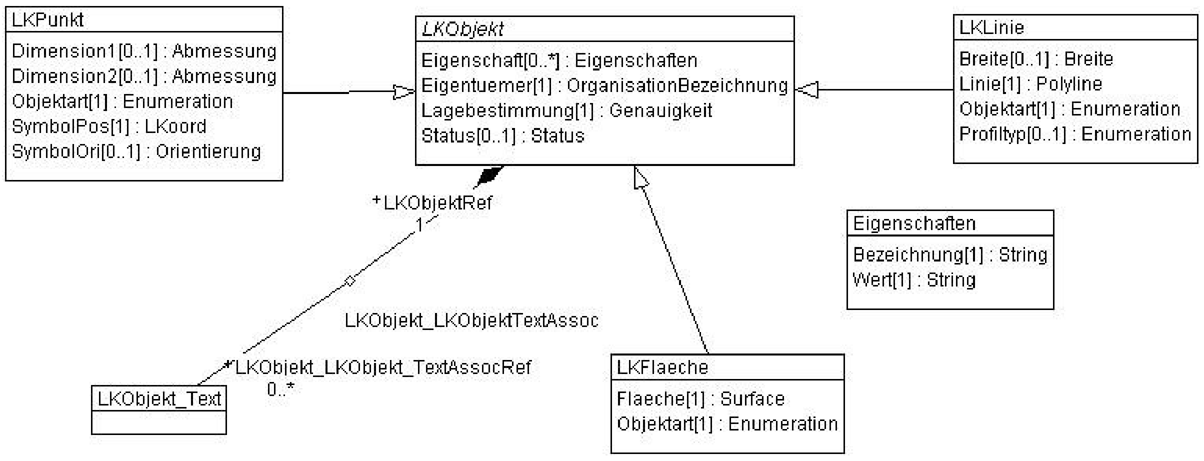
LKPunkt
Dimension1[0..1] : Abmessung
Dimension2[0..1] : Abmessung
Objektart[1] : Enumeration
SymbolPos[1] : LKoord
SymbolOri[0..1] : Orientierung
LKObjekt
Eigenschaft[0..*] : Eigenschaften
Eigentuemer[1] : OrganisationBezeichnung
Lagebestimmung[1] : Genauigkeit
Status[0..1] : Status
LKLinie
Breite[0..1] : Breite
Linie[1] : Polyline
Objektart[1] : Enumeration
Profiltyp[0..1] : Enumeration
Eigenschaften
Bezeichnung[1] : String
Wert[1] : String
+LKObjektRef
1
LKObjekt_LKObjektTextAssoc
+LKObjekt_LKObjekt_TextAssocRef
0..*
LKObjekt_Text
LKFlaeche
Flaeche[1] : Surface
Objektart[1] : Enumeration

Figure 7: UML Class Diagram Screenshot from UML/INTERLIS Editor with cardinalities

2.2.5 GeoPandas

GeoPandas, a Python library built on top of Pandas and Shapely, provides extensive support for reading and manipulating geospatial data. One of its key features is its ability to seamlessly read data from GeoPackages, a geospatial data storage format. GeoPandas leverages the geopandas.read_file() function to load data from GeoPackages, enabling users to easily access and analyze the spatial information contained within. By utilizing the GeoPandas library, users can efficiently read vector data, including points, lines, and polygons, from GeoPackages, allowing for comprehensive geospatial analysis and visualization within the Python ecosystem. The GeoPandas documentation offers detailed guidance and examples on how to utilize its GeoPackage reading capabilities for working with geospatial data [3].

GeoPandas was used throughout the import of 2D data. Easy access to GeoPackage tables and columns came as a huge benefit. The geo interface provided by GeoPandas also allowed the easy conversion from geometry in GeoPackages as BLOBs to python collections in a readable format.

2.2.6 IfcOpenShell

IfcOpenShell is a powerful open-source Python library designed for working with Industry Foundation Classes (IFC) files. It provides a comprehensive set of tools and functionalities for reading, writing, and manipulating IFC data, making it a valuable resource for BIM (Building Information Modeling) applications. With IfcOpenShell, users can parse IFC files, access and extract information about building elements, perform geometric operations, and generate new IFC files. It offers a high-level interface that simplifies the process of working with IFC data, allowing developers and researchers to leverage the library's capabilities for various purposes, including visualization, analysis, and integration with other BIM tools. The official IfcOpenShell documentation provides detailed documentation, examples, and tutorials to help users effectively utilize the library in their projects [5].

This IfcOpenShell library was used for all IFC specific functionalities. The main usage being the creation of IFC files from the given data. Generating elements based on the LKObjects provided data set, including their representations and locations.

2.3 Test Data

The focus of this thesis is data conversion. To be able to create such a conversion, data needs to be available and accessible. This project part was one of the more complicated processes, due to regulations that forbid pipe cadaster data to be publicly available.

Some test data was provided by Rothenfluh BL, Stäfa ZH and Meilen ZH. This data was analyzed with the specific focus on the LKMAP LKObjekt attribute ‘Eigenschaft’, because LKMAP does not define what kind of properties are stored in this attribute. The analysis showed that some of the provided datasets did not have the ‘Eigenschaft’ attribute. In the datasets that did have the attribute filled it was discovered that depending on the LKObjekt type the attribute was filled differently. For example, for LKLinie the attribute was filled with two key value pairs consisting of the function of the element and the usage type and LKPunkt was filled with the key value pairs for function, lid shape and visibility. These attributes were considered during the implementation of the project and are used for the element representations if they're available.

2.4 Visualizers

The following subchapters describe the evaluation done of a LKMAP viewer and various IFC viewers. The first subchapter concentrates on the LKMAP viewer used for the import data. The second subchapter describes multiple viewers used during the project for the output of generated IFC files.

2.4.1 QGIS

"QGIS is a user-friendly Open Source Geographic Information System (GIS) licensed under the GNU General Public License. QGIS is an official project of the open-Source Geospatial Foundation (OSGeo). It runs on Linux, Unix, Mac OSX, Windows and Android and supports numerous vector, raster, and database formats and functionalities." [14]

QGIS was used as an initial visualization tool for the test data provided. QGIS supports a Swiss Edition making it easy to work with INTERLIS and the LV95 Coordinate Reference System used in Switzerland.

Model Baker, a QGIS python plugin, provides further INTERLIS support by having the functionality of creating QGIS projects, PostGIS database schemas or GeoPackages from given INTERLIS models. [15]

Once INTERLIS models and data are imported into QGIS, 2D visualization of the data is presented. In the context of LKMAP it is then also possible to filter by LKMAP object type, allowing a user to only see their desired object types of the data.

However, QGIS comes with the detriment that it does not natively support IFC, the defined output format of this thesis. There is a plugin that allows IFC support in QGIS, unfortunately this plugin only supports IFC version IFC2x3. Therefore, other viewers were considered for the visualization of the output data.

2.4.2 IFC Viewers

Since during this project, the awareness was brought up that IFC viewers do not process and visualize given data in the same manner, it was decided that research would be done on the different visualization behaviors of such viewers. Another reason for this being that the resulting IFC file should be viewable in more than just one software, keeping the converter more software neutral.

The viewers looked at for this examination were LexoCAD, Blender, BIMvision and KFZViewer, experiences are described below.

The first viewer to look at for this Project was LexoCAD, which is an open-source software for 3D solutions. It’s possible to use it on its own or as an addition to cadwork. It comes with a free IFC Viewer that allows you to view valid IFC files and the properties of displayed objects. However, any modifications or exports would require a license.

Using LexoCADs IFC Viewer is great as long as you are sure that your IFC file is valid, otherwise there may be some confusion due to the lack of warning and error messages provided by the software. [16]

Once the decision was made that the IfcOpenShell library was going to be used in the project, BlenderBIM came up as an IFC Viewer option. BlenderBIM Add-on is as its name suggests an addition to the free and open-source 3D creation suite Blender. The IfcOpenShell documentation even suggests using BlenderBIM as a graphical interface when using the library. [17]

Even so BlenderBIM was not the best option for this project. It very quickly led to using BlenderBIM specific functionality in IfcOpenShell for simplicity, without the realization of the consequence that the generated IFC file would only be viewable in Blender.

BIMvision was also looked at as a probable viewer for the generated IFC files. However, BIMvision does not officially support IFC 4.3 which is why it was eventually disregarded. [18]

The final viewer, FZKViewer, came up as an interesting consideration due to the fact that it supports the visualization of semantic data models for BIM and GIS. Unfortunately, the FZKViewer also does not support the IFC 4.3 version. [19]

Taking these experiences into consideration, the project focused on viewing generated data in LexoCad and optimizing it for this choice.

3 Solution Concept and Results

Due to this project containing diverse facets of data states and data formats, the solution strategy was divided into four parts. For each of these parts a separate POC (Proof of Concept) was established.

3.1 Interactions with Command Line Interface

COMMAND LINE INTERFACE	
ANALYSIS	A command line interface should be simple with the possibility for more complex input if desired by the user. Therefore, an analysis was done on what required user arguments are needed and which arguments would be nice to have for a user.
POC	The idea is to create a CLI that requires the user to input a LKMAP data file (.xtf) and the corresponding model file(.ili). Due to the large amount of data objects provided in such an LKMAP data file and the consideration that IFC BIM files are meant to represent one site, it was decided that the user should also be required to input a coordinate range (-*clipsrc* [xmin ymin xmax ymax]). This range creates the area to be displayed in the resulting output file.
RESULT	The resulting CLI allows the user to input a GeoPackage, a coordinate range for the objects and the path to where the IFC file should be generated. It validates that all the arguments are given, that the GeoPackage can be found and that the IFC file does not already exist. Executing the command 'convert-gpkg-to-ifc' with the abovementioned arguments the result is a valid IFC 4.3 file that can viewed in a BIM viewer. The POC was not successfully validated, because the arguments taken are not LKMAP data files and LKMAP model files, but instead GeoPackages. This comes as a consequence of the second solution strategy discussed in the next subchapter not being successful either.

Table 4: CLI Solution Strategy

3.2 Importing and Processing Data

IMPORTING AND PROCESSING DATA	
ANALYSIS	The analysis done for this first step was used to define a concept that could later be proved as usable. For the elaboration of this con-

	cept it was necessary to gain knowledge about the predefined formats and standards and the various libraries that could potentially achieve the wanted functionality. These evaluations can be found in Chapter 2.
POC	The concept to prove was to successfully be able to convert INTERLIS transfer data into a GeoPackage using Ili2GPKG in python and then accessing this data with the GeoPandas library.
RESULT	Because the concept of converting ILI2GPKG to a native image with GraalVM was let go the CLI requires a GeoPackage instead of the originally planned LKMAP data. The outcome of importing data therefore included a FilterChain consisting of filters applied to the given GeoPackage. The first of these filters minimizes the number of data objects with the coordinate range CLI argument, the next filter groups these into their respective LKObjekt types and the final filter adds characteristic attributes to the objects.

Table 5: Data Import Solution Strategy

3.3 Adding Default Data

ADDING DEFAULT DATA	
ANALYSIS	Considerations needed to be made as to how the data accessed in the step before will be temporarily stored. How this data can be / needs to be enhanced for proper 3D visualization and which approach should be used for modelling in the next step.
POC	This proof of concept defines that given data should be enhanced with default values for properties that are needed for proper 3D visualization.
RESULT	This POC was successful and implemented accordingly. "Intelligent Defaults" enhance the provided data with altitudes, widths of elements, color schemes and scaling to set the origin point of the coordinate system to the minimum values of the provided range

Table 6: Default Data Solution Strategy

3.4 IFC Export

Coming to exporting the data and then visualizing it the question arises which IFC Elements / Entities should be used. The first subchapter goes into detail on the difference between a concrete or abstract approach when it comes to IFC Elements / Entities.

The second subchapter then describes the POC that was defined following the analysis of the first subchapter.

3.4.1 Concrete or Abstract Approach

The reasoning behind the discussion of a concrete or abstract approach in IFC comes from the fact that LKMAP is an abstract minimalistic model (Detailed in Chapter 2.1.2), consisting of points, lines, and areas. Logically it's not possible to visualize these elements as 3D objects, therefore the question arises what kind of IFC element should be used to display these different types.

The diagram on the right visualizes the different IFC elements considered for this project.

IfcBuildingElementProxy is the most abstract element considered. It's a subtype of IfcBuiltElement which contains all primary construction elements of a built facility, and it has no predefined meaning of the type of element it is representing and therefore called a proxy. [20]

A little more concrete is the IfcDistributionFlowElement. Inheriting from the IfcDistributionElement it's defined as a participant in a distribution system and the flow aspect of the element categorizing it as an element that facilitates distribution in contrast to its counterpart IfcDistributionControlElement which is an element controlling said flow. [21]

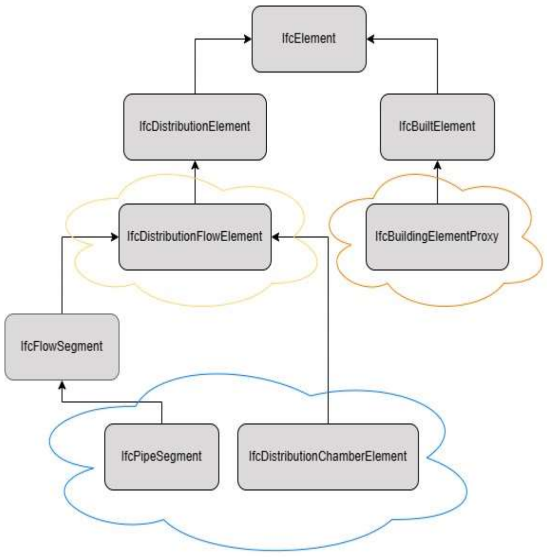

Figure 8: IfcElement Types Diagram

The concrete elements considered are displayed at the very bottom of the diagram. Consisting of IfcDistributionChamberElements which would be used to display ducts and IfcPipeSegments which would be used to display pipes. It's worth noting that the concrete elements listed are not all the possible elements one could use but the ones considered in this project.

As discussed at the top of this subchapter, LKMAP is an abstract approach, which suggests it would be a good option to use either one of the abstract approaches over the concrete one, to keep the abstraction level for output data in line with the abstraction level of the input data. Also coming with the advantage being that the implementation would be simpler, by not having to differentiate the LKObjekt type when building an IfcElement. However, there was a disadvantage found concerning the property sets that are predefined in IFC, but these property sets were deemed out of scope for this project, resulting in this detriment having minimal to no impact.

Between the two abstract element types IfcBuildingElementProxy and IfcDistribution-FlowElement it was decided to use the latter. This decision was made because LKMAP is indeed abstract but also only used in a pipe cadaster setting. IfcDistributionFlowElement was and is the better fit due to it also being for distribution systems like LKMAP.

In conclusion, the IfcElements considered were compared to each other and to LKMAP to achieve the best possible fit, while also considering restrictions or other disadvantages that could occur after a decision was made. The final decision was based on the guideline to use an abstract approach that includes some concrete aspects that match those of LKMAP.

3.4.2 Exporting data and Viewing

ANALYSIS	The concept for this step was elaborated by gaining knowledge about the output format specified in the problem space (IFC Chapter 2.1.4), evaluating libraries that support writing IFC files and assessing which visualization tool to use for then displaying the exported data in 3D (Visualizers Chapter 2.4).
POC	The concept to prove was to write an IFC 4.3 file with the IfcOpenShell library that is valid and contains enough information to be able to visualize its elements in a commonly used visualizer.
RESULT	The result contains IfcElementBuilders using IfcOpenShell that create the elements from the processed data and allocate these to a created IFC 4.3 file.

Table 7: Exporting Data Solution Strategy

4 Summary and Outlook

In summary, three of the four solution strategies were successfully completed, resulting in a CLI prototype that allows the user to input a GeoPackage, a coordinate range for the objects and the name of the IFC file to be generated. Executing the command convert-gpkg-to-ifc with the above-mentioned arguments the result is a valid IFC 4.3 file ready to be visualized with an external IFC viewer program, like LexoCAD, Blender or BIMvision. With data processing filters, the number of data objects is minimized by filtering out objects which do not intersect with the range or specifically the bbox (boundary box), objects are grouped into their LKObjekt types, and characteristic attributes are added to the corresponding object. In addition, "Intelligent Defaults" enhance the provided data with location heights, element widths, color schemes, and scaling to set the origin of the coordinate system to the minimum values of the provided bbox ensuring that the bottom left corner of the bbox is the (0,0,0) coordinate point. Finally, IfcElementBuilders using IfcOpenShell create the elements from the processed data and assign them to a generated IFC 4.3 file.

With this foundation set, there are various improvements to be implemented. The usability of the prototypes generated IFC file can be improved by marking default data, in order to make certain that users are not led to believe that all the data is accurate. Intelligent Defaults can be enhanced with flow direction, altitude gradients and further characteristics for more probable results. Another enhancement would be to have functionality to support the creation of IFC files with the other LKMAP utility network mediums. Once those mediums are fully supported there is also the possibility to create functionality to be able to load specific medium models that contain more attribute and property information for the elements.

Further possibilities include the functionality to provide the user with configurability for intelligent defaults or the functionality for users to add additional LKMAP data to existing IFC files.

PART II

1 Vision

As described in Part I Chapter 1

2 Requirements

This section lists all functional and non-functional requirements and the main use case this application seeks to cover. These requirements were further used as a guideline for architectural, design and implementation decisions.

2.1 Use Case

Actor:

Civil Engineer, Architect, etc.

Wants:

Conversion form LKMAP 2D data to IFC 3D Data with visualization

Condition:

User has a GeoPackage database file and a coordinate range

Scenario:

1. The user opens the command line interface.
2. The user configures the conversion command with the required arguments.
3. The user runs the conversion command.
4. The application validates the given arguments and input dataset.
5. The application parses the input dataset, filtering the relevant data, including border restrictions via a BBOX.
6. The application enhances the parsed data with default values/smart assumptions for missing data parameters and denotes these enhancements as assumed data.
7. The application maps the enhanced data to appropriate IFC entities and properties, including a differentiable representation for assumed data.
8. The application generates a valid IFC file with the created IFC entities.
9. The application saves the IFC file with the file name provided by the command line argument given by the user.
10. The application asks the user with which viewer the generated IFC file should be opened, for optimal inspection.

2.2 Functional Requirements

The first 5 functional requirements define the general requirements of the whole application. The following functional requirements are grouped by their component.

GENERAL FUNCTIONAL REQUIREMENTS	
CLI	The application should provide a command-line interface that allows users to interact with the software through text-based commands
DATA PARSING	The application should parse the input 2D data set, extracting relevant information such as geometry, attributes, and properties of elements.
DATA ENHANCING	The application should enhance given data with necessary properties based on smart assumptions, to ensure that the output is valid and represented in a user-friendly fashion
IFC FILE GENERATION	The application should generate a valid IFC file containing the converted 3D dataset, adhering to the IFC file format specifications.
ERROR HANDLING	The application should handle errors and exceptions gracefully, providing informative error messages to users in case of invalid input, parsing issues, or other conversion errors.
CLI FUNCTIONAL REQUIREMENTS	
INPUT VALIDATION	The module should validate the input data set to ensure it is in a compatible format and meets the required specifications for conversion to an IFC file.
ARGUMENT VALIDATION	The module should validate that the user has given all required arguments in the CLI
NO OVERWRITE VALIDATION	The component should validate that the created IFC file does not already exist, in order to not overwrite that file
DATA IMPORT REQUIREMENTS	
RANGE RESTRICTION	The module should filter all Objects within the given Range
RANGE COORDINATE OVERFLOW	The module should validate that only complete Objects are available. Meaning if there is coordinates outside and inside of the given range of an object, all of the coordinates are given (incl. coordinates outside the given range)
INTELLIGENT DEFAULT REQUIREMENTS	
ALTITUDE/HEIGHT DEFAULT	The module should enhance the given data with the 3D coordinate Z, used for altitude/height and denote that this enhancement is an intelligent default

COLOR ENHANCEMENT	If not already present in the data, the module should enhance the objects with assumed material and denote that this enhancement is an intelligent default
IFC FUNCTIONAL REQUIREMENTS	
DESCRIPTION	The module should validate the generated IFC file to be valid
SAVING IFC FILE	The module should be able to save the validated IFC file to a given location provided by the user
IFC VIEWER	The module should allow the user to open an application used to view the generated IFC file

Table 8: Functional Requirements

2.3 Non-Functional Requirements

The below table lists all project relevant non-functional requirements. The Non-Functional Requirement Context Security is not defined due to it not having any relevance/necessity in this project.

NFR Context	Description	Acceptance Criteria
Usability	The application must be user-friendly and intuitive	-- help argument for CLI that lists all argument possibilities and gives the argument format
Portability	The application must be portable and able to run on at least Windows and Linux devices without requiring significant modification.	Manual Testing on Linux and Windows devices
Compatibility	The application must support the INTERLIS v2.3 and IFCv4.3 standards	Unit Tests
Availability	The application must be open source (if possible with a permissive license).	Application has permissive license
Code Quality	The application must be written in easy to understand clean python code, allowing easy maintainability, PEP8	Pipeline with Qodana-Python Linter is run after every commit

Table 9: Non-Functional Requirements

3 Architecture

This chapter focuses on the architectural decisions made for this project.

3.1 Context Diagram

The GIS-to-BIM Converter Prototype interacts with a user over the provided command line interface and writes generated data to the file system of that user. Therefore, the System Context shown in the diagram below, following the C4 Diagram Notation, is simple and easy to understand.

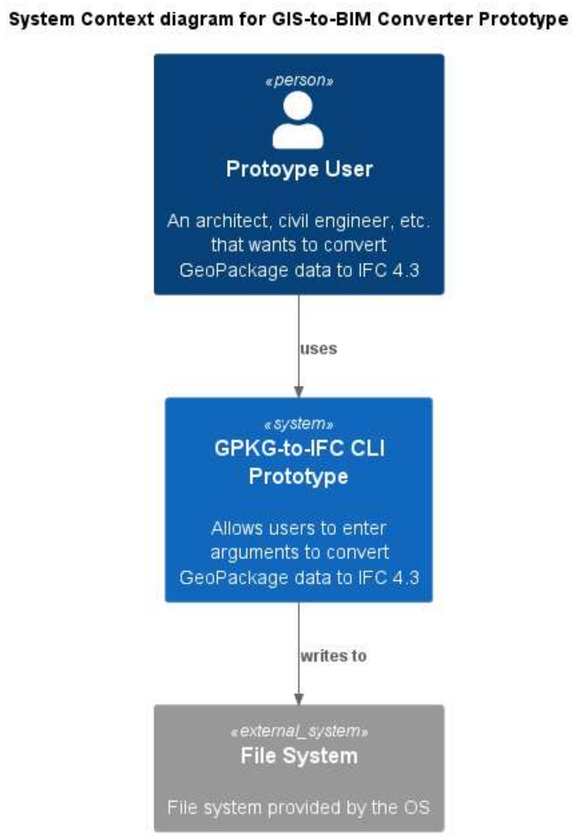
System Context diagram for GIS-to-BIM Converter Prototype
«person»
Protoype User
An architect, civil engineer, etc. that wants to convert GeoPackage data to IFC 4.3
uses
«system»
GPKG-to-IFC CLI Prototype
Allows users to enter arguments to convert GeoPackage data to IFC 4.3
writes to
«external_system»
File System
File system provided by the OS

Figure 9: System Context Diagram

3.2 Containers

Taking a look into the system, it is split into two containers, namely the Command Line Interface container and the Data Processing Controller container. The CLI container uses the Python Typer which could be exchanged for any other CLI library. The controller contains all the business logic and data processing steps done to create a valid IFC 4.3 file. The components of the Data Processing Controller are described in detail in the next section (Chapter 3.3)

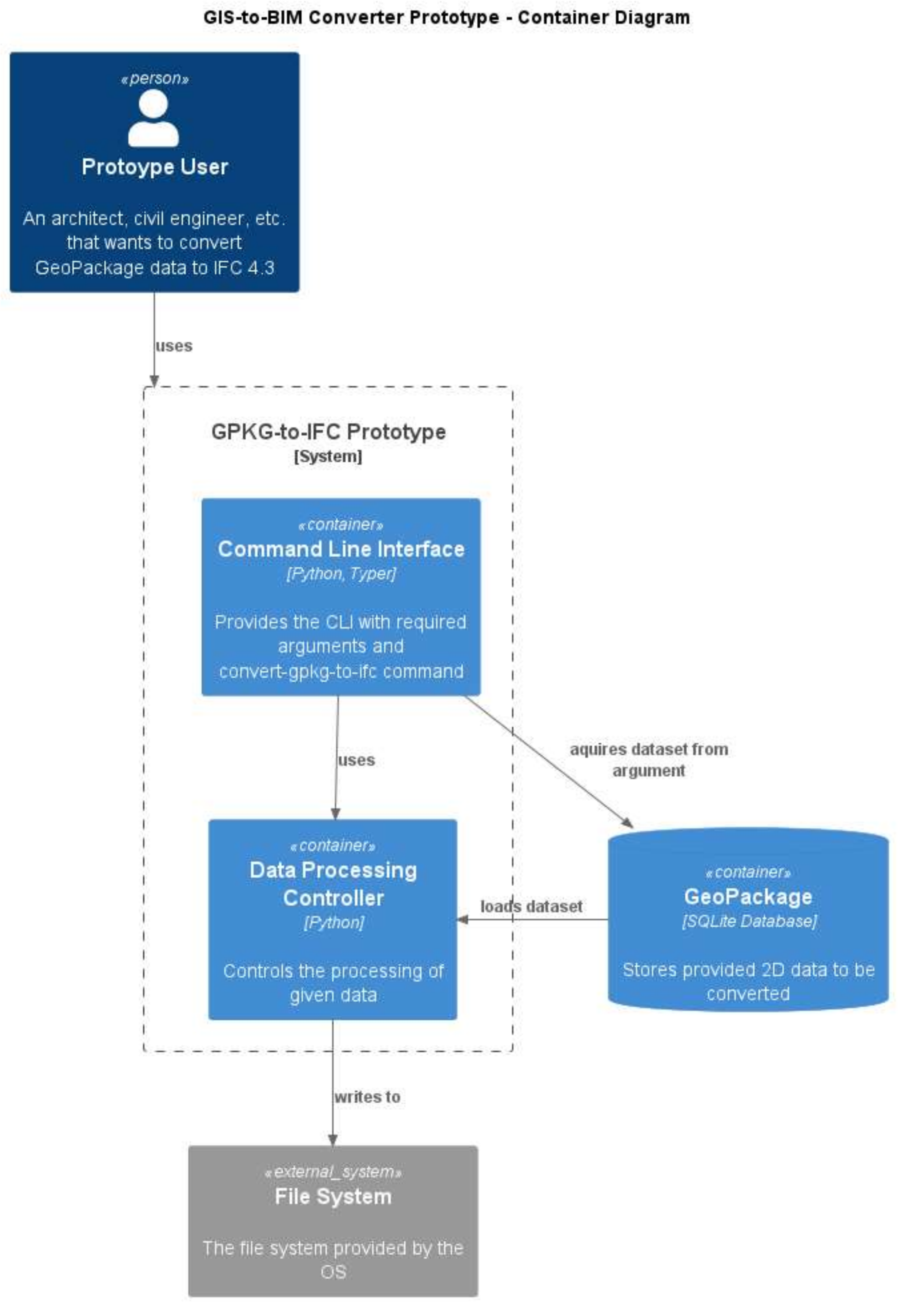
GIS-to-BIM Converter Prototype - Container Diagram
«person»
Protoype User
An architect, civil engineer, etc. that wants to convert GeoPackage data to IFC 4.3
uses
GPKG-to-IFC Prototype
[System]
«container»
Command Line Interface
[Python, Typer]
Provides the CLI with required arguments and convert-gpkg-to-ifc command
uses
aquires dataset from argument
«container»
Data Processing Controller
[Python]
Controls the processing of given data
loads dataset
«container»
GeoPackage
[SQLite Database]
Stores provided 2D data to be converted
writes to
«external_system»
File System
The file system provided by the OS

Figure 10: Container Diagram

3.3 Components

This subchapter focuses on the Components of the Data Processing Controller. The architectural decisions made for each component are described below the Component Diagram which shows an overview of the relationships between the components.

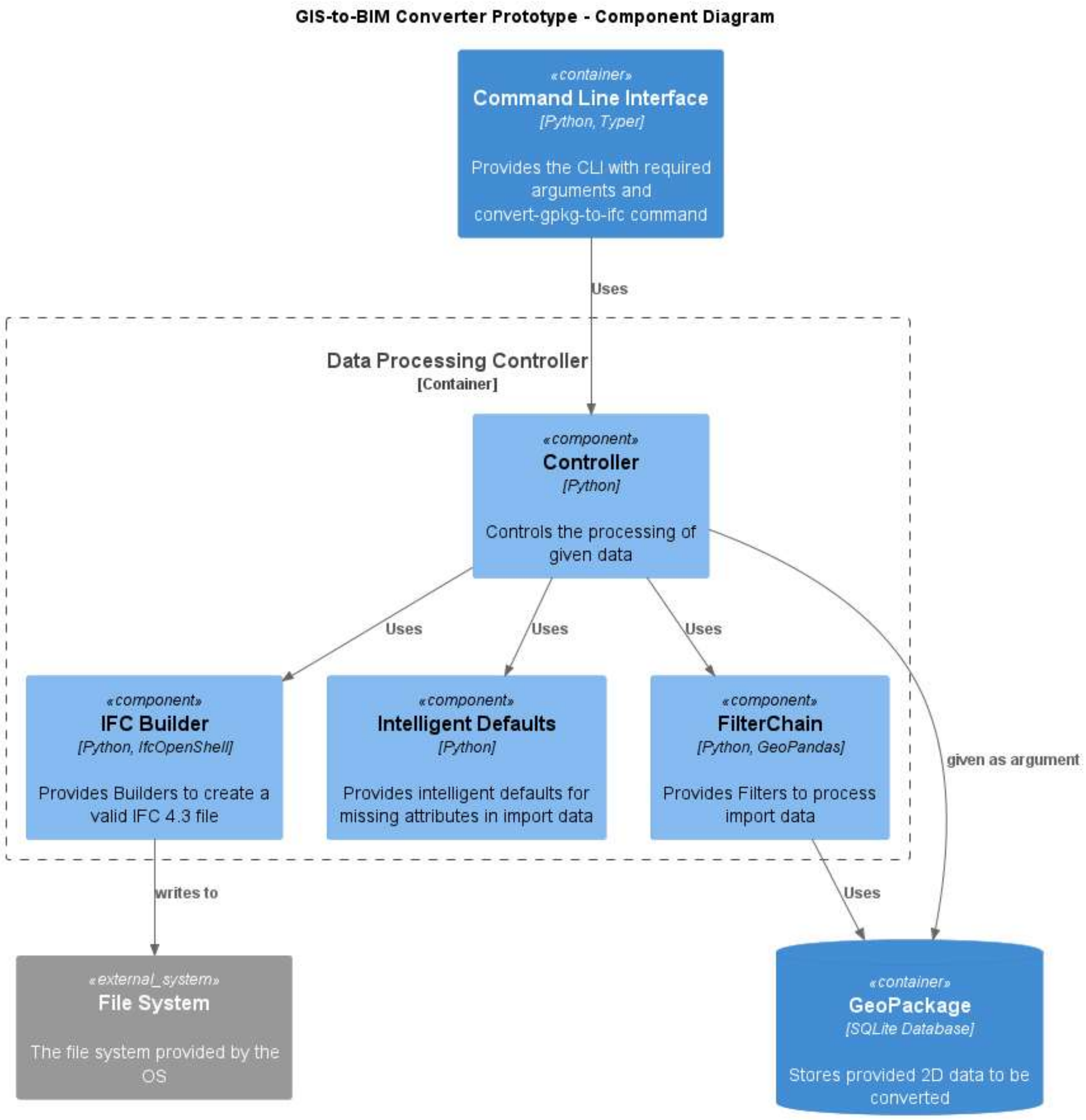
GIS-to-BIM Converter Prototype - Component Diagram
«container»
Command Line Interface
[Python, Typer]
Provides the CLI with required arguments and convert-gpkg-to-ifc command
Uses
Data Processing Controller
[Container]
«component»
Controller
[Python]
Controls the processing of given data
Uses
Uses
Uses
given as argument
«component»
IFC Builder
[Python, IfcOpenShell]
Provides Builders to create a valid IFC 4.3 file
«component»
Intelligent Defaults
[Python]
Provides intelligent defaults for missing attributes in import data
«component»
FilterChain
[Python, GeoPandas]
Provides Filters to process import data
writes to
Uses
«external_system»
File System
The file system provided by the OS
«container»
GeoPackage
[SQLite Database]
Stores provided 2D data to be converted

Figure 11: Component Diagram

Controller

The controller acts as the supervisor of all other data processing steps to be done. It uses each of the other components and is given the dataset from the CLI, which is later used by the FilterChain component.

FilterChain

As the name suggests the FilterChain component consists of a number of filters that execute the first processing steps of the given dataset. The structure of the classes within the component was built on the Intercepting Filter Pattern.

In the context of the FilterChain component facing the concern of data processing the given GeoPackage, it was decided to go with the option of the Intercepting Filter Pattern and neglect the option of the iterator pattern, to achieve performant and decoupled steps of data processing, accepting the downside of not knowing the order or current position of elements.

Intelligent Defaults

The Intelligent Defaults component is currently split into two classes, one that is used for scaling the zero point of the model and the other for defined color defaults.

In the context of having not enough data in the 2D dataset facing the concern of 3D visualization it was decided to go with Intelligent Defaults and neglect other options like smart look ups, to achieve a simple data enhancing approach, accepting the downside of the intelligent defaults possible inaccuracies.

IFCBuilder

The last component that is called from the controller would be the IFCBuilder. Internally it is structured according to the Builder Pattern.

In the context of the IFCBuilder component facing the concern of creating IfcElements that can be viewed in a BIM viewer, it was decided to use the Builder Pattern neglecting the option of the Factory Method Pattern to achieve complex object creation, accepting the down side of lack of compile-time safety.

4 Implementation & Testing

This chapter looks at some of the interesting aspects encountered during the implementation of this project and describes the testing methodology used.

The implementation process for each component was different. For the CLI the process was kept simple, by finding a Python library that would support a simple CLI with arguments. The library chosen was Typer [22], for its easy readability and maintainability.

For the data processing component FilterChain the implementation process was done through discussing what filtering steps the GeoPackage would have to go through to get to a dataset that can then be used for the IFC Builders. Figure 12 below shows the steps that are taken in the FilterChain. The hexagons display what attributes are given to the filter and the colored rectangles present which data format the dataset has after the filter is executed.

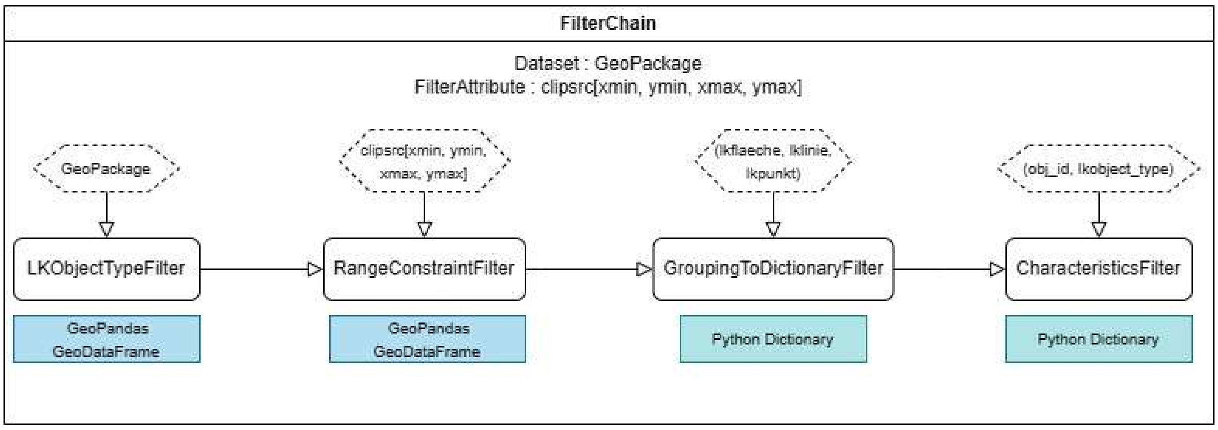

Figure 12: FilterChain Diagram

The testing for this component was done with standard unit tests, some of which contain mocks of the GeoPandas GeoDataFrame.

Implementing the IFC Builder component was done with a Test Driven Development (TDD) approach. Because IFC has various options of how to build representations for elements, different tests were set up to find the optimal representation for each of the element types. Once this optimal representation was found the implementation of IFC Builders began following a component design using the Builder Pattern.

Test Data

A lot of the testing described above required a valid test dataset. The dataset used was from Rothenfluh, BL and within that municipality it was decided to choose a coordinate range that would include a building and a road intersection. This led to using the coordinate range (2’635’955.3, 1’256’666.5, 2’635’997.8, 1’256’709.9) which creates a bounding box around the address Anwilerstr. 10 in Rothenfluh.

5 Results & Forecast

This chapter describes the results of the project compared to the requirements defined at the beginning phases. Further it lists the possible opportunities to enhance and further the development of this prototype.

Result

Overall, the resulting prototype covers a lot of the Non-Functional Requirements (NFR Chapter 2.3) defined. The NFR for portability is a requirement that was not fully tested and should be proved as accepted in the future of this prototype.

Another improvement could be made for the usability context, it could use some fine tuning when it comes to the argument descriptions provided by the –help command to ensure there are no problems when trying to convert a GeoPackage.

Finally, the resulting prototype in its current state proved that a NFR that was not defined at the beginning of the project but could be added would be performance. The demo that was done using this prototype had a considerably better performance than originally suspected, even if the coordinate range filter which minimizes the amount of data is left out.

In the Functional Requirement (FR) context (Chapter 2.2) almost all requirements are covered in the current project state. The FR for handling range coordinate overflow is handled through using the cx method provided by GeoPandas which ensures that all objects that intersect with the bounding box are taken, this was tested in a unit test.

The resulting prototype could use some improvement when it comes to FRs that require data or file validations. In the current state there is no validation that the IFC file is valid, the user will only know if it is once he opens the model in a viewer.

Forecast

The project in its current form provides a good foundational prototype for further development and provides a nice demonstration of what can be improved and added.

Facing the fact that the goal of using a native image of the ILI2GPKG library was not fulfilled in this project, the main outlook is to complete this goal by fully proving the defined POCs and integrating these concepts into the existing prototype, which will happen as part of the follow-up bachelor thesis.

There is also the possibility to let go of the native-image idea and implement a custom xml parser that could parse the given LKMAP data. However, this idea comes with a large time investment and analysis to make sure that it is compatible with multiple models, which is the end goal for this system.

The prototype does need some improvement, especially when it comes to displaying the intelligent defaults. To avoid having users assume that the data in the IFC file is accurate it will be a requirement in the following bachelor thesis to mark intelligent defaults as possibly inaccurate.

The Intelligent Defaults themselves should be enhanced with flow direction and altitude gradients to ensure that the model does not present impossible situations, like the current problem of pipes crossing each other at the same level. In this context it would also be nice to have the functionality of providing the user with configurations so that the Intelligent Defaults can be set to the users liking. This would ensure that the user is more aware of the fact that there is an inaccuracy tolerance in the resulting model.

The possibilities of how this prototype can be enhanced further are endless. However, the next step that will be taken for following bachelor thesis is supporting the other mediums present in the LKMAP model. Further, the goal can be expanded to the specific models behind LKMAP, which do not follow such a minimalistic approach.

The questions that arise when considering the outcome of this project are if there is a way to ensure that Intelligen Defautls are as accurate as possible? How was the depth of the elements handled so far, if the data is not in the original model? What are the users most desired requirements for further implementations?

APPENDIX

Glossary

Term	Explanation
BIM	BIM is short for Building Information Modeling, a process of creating and managing digital representations of physical structures in 3D.
GIS	GIS is a classic Geo Information System. Spatial data is supplemented with factual Data. Comprehensive models also show relationships between various layers, for example which hydrant is connected to which water pipe.
IFC	An IFC file is a model file created in the Industry Foundation Classes (IFC) formats, an open file format used by Building Information Modeling (BIM) programs. It contains building or facility models, including spatial elements, materials, and shapes. It is used to facilitate interoperability and collaboration between different software applications using IFC-based capabilities. IFC files are essential for data exchange in BIM.
INTERLIS	INTERLIS is a Swiss model-based data description language and exchange format. In particular, data models from public sectors are described in this format for example spatial planning data or official cadastral measurements.
LKMAP	LKMAP is the standard defined by the SIA to transport information on the cadastral register form one system (particularly GIS) to another – using INTERLIS.
Pipe Cadaster	Pipe Cadasters include information about what is where in the ground (Water pipes, cables, sewers, shafts, etc.) Usually, this data contains coordinates and possible depth below ground. However, the exact altitude is rarely provided.
SIA 405	The SIA 405 is a standard that applies to the exchange and publication of network information data and pipeline cadasters. It defines the minimum requirements of the procedures for GIS-based documentation of public and private pipes and related installations intended for distribution and sanitation.
QGIS	QGIS is a user friendly Open Source Geographic Information System (GIS) licensed under the GNU General Public License. QGIS is an official project of the Open Source Geospatial Foundation (OSGeo). It runs on Linux, Unix, Mac OSX, Windows and Android and supports numerous vector, raster, and database formats and functionalities.

Table 10: Glossary

Bibliography

[1] "SIA 405," [Online]. Available: https://www.sia.ch/de/dienstleistungen/normen/themen/geodaten/. [Accessed March 2023].

[2] "Ili2db," [Online]. Available: https://www.interlis.ch/downloads/ili2db. [Accessed March 2023].

[3] "GeoPandas," [Online]. Available: https://geopandas.org/en/stable/index.html. [Accessed April 2023].

[4] "GraalVM - Native-Image," [Online]. Available: https://www.graalvm.org/reference-manual/native-image/. [Accessed June 2023].

[5] "IfcOpenShell," [Online]. Available: https://ifcopenshell.org/. [Accessed March 2023].

[6] INTERLIS. [Online]. Available: https://www.interlis.ch/. [Accessed March 2023].

[7] "GeoPackage," [Online]. Available: https://www.geopackage.org/spec/. [Accessed May 2023].

[8] "Building Smart - IFC," [Online]. Available: https://www.buildingsmart.org/standards/bsi-standards/industry-foundation-classes/. [Accessed March 2023].

[9] [Online].

[10] "Ili2fme," [Online]. Available: https://www.interlis.ch/downloads/ili2fme. [Accessed March 2023].

[11] "GraalVM - Native-build - Issue," [Online]. Available: https://github.com/graalvm/native-build-tools/issues/318.

[12] "Ogr2ogr," [Online]. Available: https://gdal.org/programs/ogr2ogr.html. [Accessed March 2023].

[13] "INTERLIS - umleditor," [Online]. Available: https://www.interlis.ch/downloads/umleditor. [Accessed March 2023].

[14] "QGIS," [Online]. Available: https://qgis.org/de/site/. [Accessed March 2023].

[15] "Model Baker," [Online]. Available: https://opengisch.github.io/QgisModelBaker/. [Accessed May 2023].

[16] "LexoCAD," [Online]. Available: https://07.cadwork.ch/index.php/en/. [Accessed May 2023].

[17] "BlenderBIM," [Online]. Available: https://blenderbim.org/docs/users/installation.html. [Accessed June 2023].

[18] "BIMvision," [Online]. Available: https://bimvision.eu/de/. [Accessed May 2023].

[19] "KIT," [Online]. Available: https://www.iai.kit.edu/1648.php. [Accessed May 2023].

[20] "IfcBuildingElementProxy," [Online]. Available: https://ifc43-docs.standards.buildingsmart.org/IFC/RELEASE/IFC4x3/HTML/lexical/IfcBuildingElementProxy.htm. [Accessed May 2023].

[21] "IfcDistributionFlowElement," [Online]. Available: https://ifc43-docs.standards.buildingsmart.org/IFC/RELEASE/IFC4x3/HTML/lexical/IfcDistributionFlowElement.htm. [Accessed May 2023].

[22] "Typer," [Online]. Available: https://typer.tiangolo.com/. [Accessed June 2023].

[25] [Online]. [Accessed March 2023].

[26] "CRB Digital Twin Oberegg," [Online]. Available: https://www.crb.ch/Stories/Digital_Twin_Oberegg.html. [Accessed May 2023].

[27] "GDAL," [Online]. Available: https://gdal.org/. [Accessed March 2023].

Table Of Figures

Table of Tables