Code-Manual

Research project EVER II

Prof. Dr. Mirjam Minor

Georgios Brussas

Julius Rubbe

www.wi.cs.uni-frankfurt.de

Manual

This manual describes and explains the code of the EVER II project. The manual is based on the original code by Xinyuan Cai and includes all extensions that followed. The goal of the manual is to provide new project participants with a quick overview of the code structure, as well as the content. The complete program code can be found in Gitlab (link: http://141.2.11.167/gitlab/minor/ever-tlframework). There are also detailed comments explaining the implementation of the code in more detail.

It is a project to extract, generalize and abstract workflows from an ontology of BPMN workflows for the purpose of transferring procedural knowledge between two domains. The code can be executed in a Java 8 environment.

The code was created in version 4.16 of the Eclipse project, which is developed on Java SE 8 VMs. Therefore, the Eclipse SDK as a whole is tailored to all modern desktop Java VMs.

Overview

1. Structure of the code

First, the structure of the present code is explained to allow a quick classification of the individual files. It is helpful to look at the structure of the code, which can be found in Figure 1. The most important files can be found in the areas "evaluation", "src/main" and "target". In addition, an overview of all Java files within the folders can be found in Figure 2. The figures are taken from the PowerPoint file "Hierarchie_Code_EVERII, which was uploaded to Gitlab.

Files structure from Xinyuan:

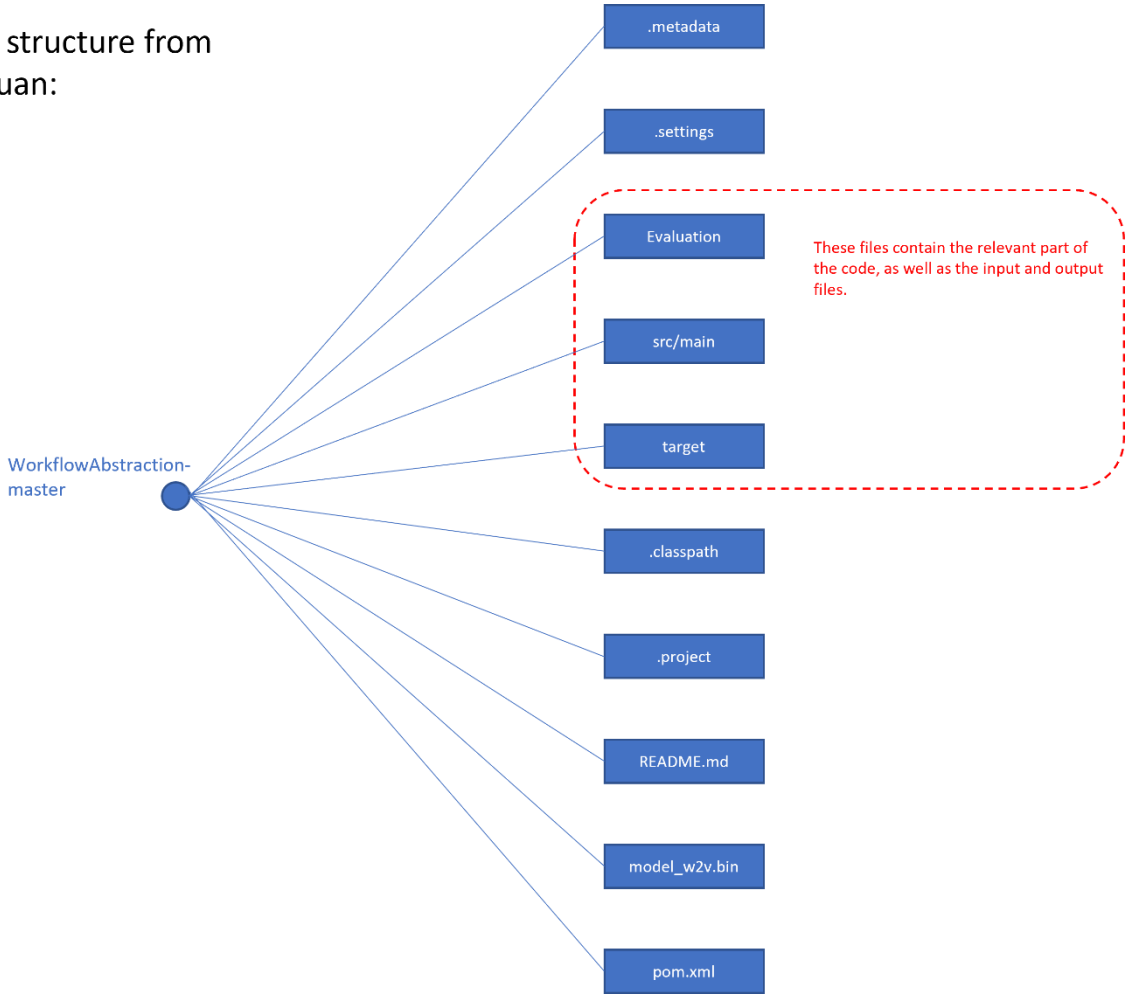


Figure 1: Overview over all Folders

Figure 1 shows all folders of the code structure. There are only three relevant folders for further programming. These are "evaluation", "src/main", and "target". These will be described further in the following. The other files are uninteresting for the time being.

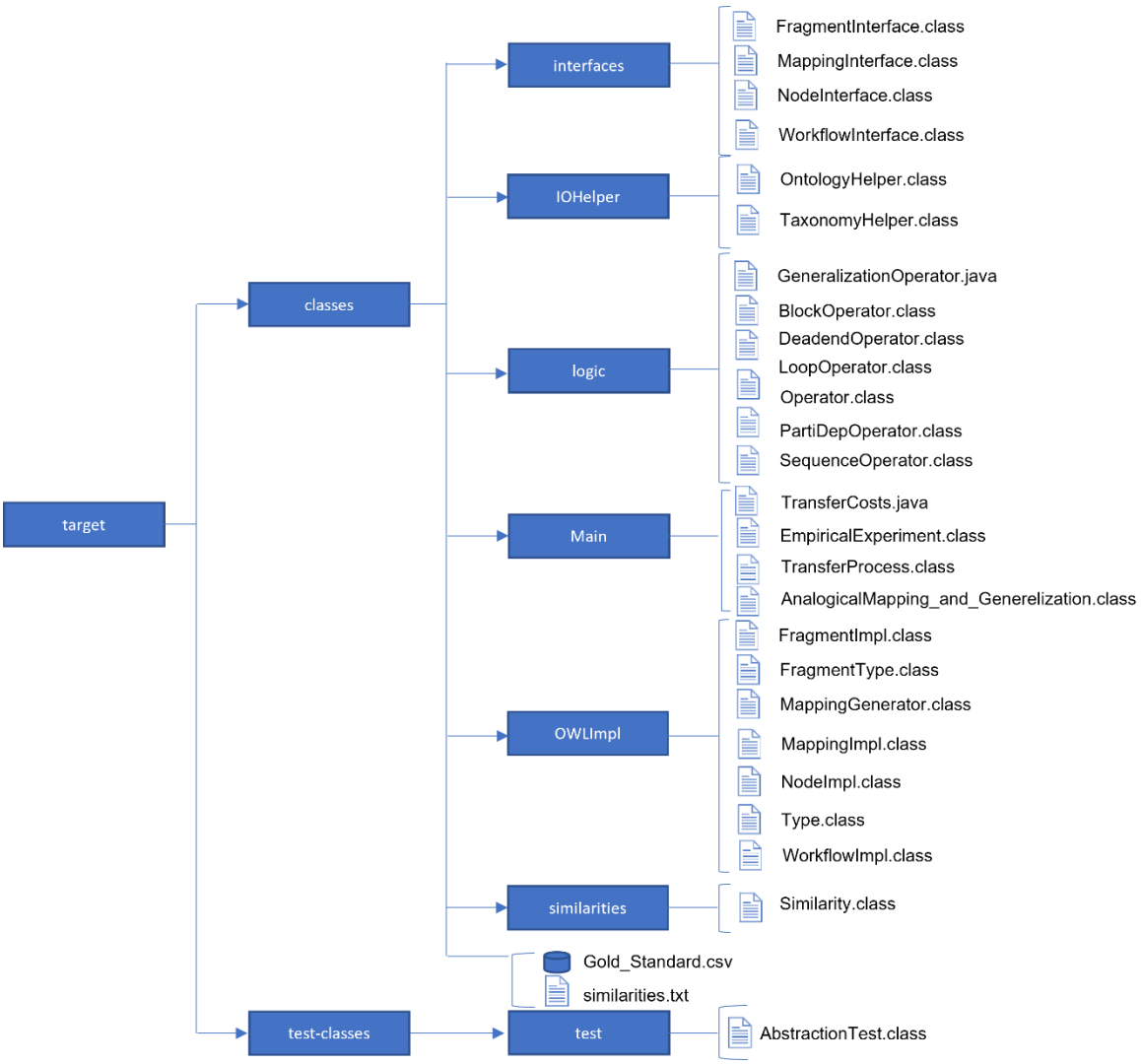


Figure 2: Overview Java-files

Figure 2 maps all Java classes in the code. The most relevant ones are explained in more detail in the second chapter. In general, many classes are combined with each other. In the upper part of the respective class can always be looked, which other class is imported. This is especially relevant for further transfer operators, because not everything has to be reprogrammed, but files like the TaxonomyHelper can be imported easily.

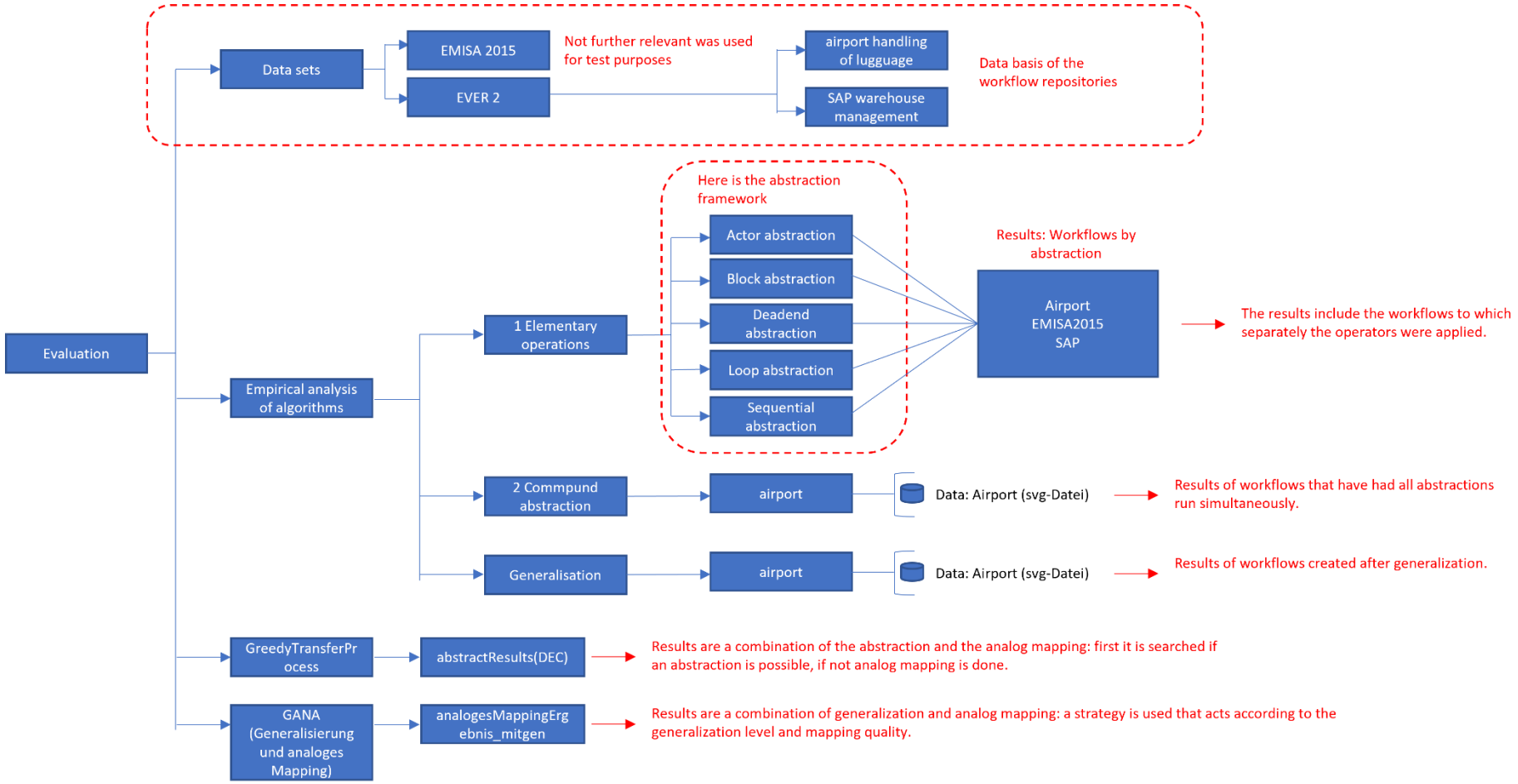


Figure 3: Overview Input- und Output-files

Figure 3 shows, among other things, the input files for the project. These are located in the Data sets folder, EVER 2, where the data from the two domains are stored. EMISA 2015 is only a test file. Empirical analysis of algorithmus is one of the most important files in the code because it is

where the output is stored. Elementary operations contains the results of each abstraction operator, while Commpund abstraction contains the results of all abstraction operators on a workflow. The Generalization folder stores the results after a generalization.

Another folder in the file structure is GreedyTransferProcess, which mixes abstraction and analog mapping and performs a transfer transfer from the source to the target domain. The results are stored in the abstractResults(DEC) folder.

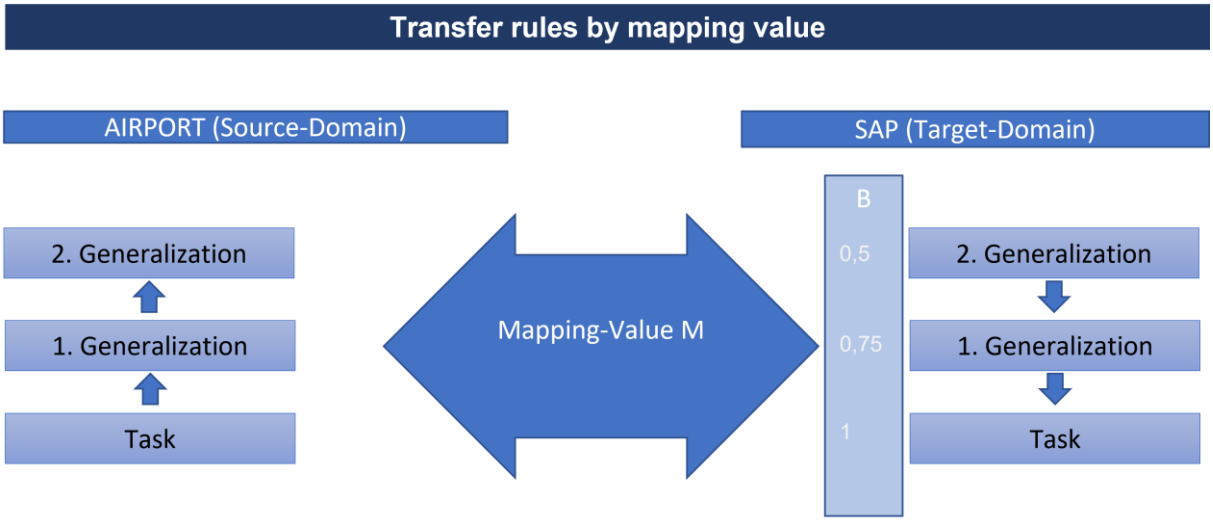
The results from GANA, a combination of generalization and analog mapping are stored in the folder "analogMappingResults_mitgen". A transfer strategy is used here, which is also described in chapter 1.1.

2. Wichtige Klassen des Codes

➔ In the following, the most important classes are outlined and described. It should become clear how the classes are interrelated and intertwined.

1.1 AnalogicalMapping_and_Generalization

This Java class is found in the main file. It is responsible for a transfer with analog mapping and generalization. The program works with an algorithm that optimally selects whether and how the transfer takes place. This is done by a mapping value. The idea is that if a task is generalized, then the transfer is less attractive. Therefore, the level of transfer is considered and a mapping value is assigned to each level. An overview of this can be seen in the following graphic:



M = K * A * B
with confidence value K, transfer side A and transfer side B.

(Confidence for task task is Greedy calculation of Nils (COS-SIM) and for manual mapping (Ontology Engineering) at 1) Choose for each task in airport the transfer path with the highest mapping value M

2.1 Empirical Experiment (Main-Methode)

This Java class is located in the main files and is one of two files that generates an output. Thereby the output files for the abstraction operator are created. Among other things, it provides the results of the individual abstractions, as well as the results that result from the combination of the various abstraction operators.

2.2 TransferProcess (Main-Methode)

TransferProcess calls the files which are necessary for the generalization operator. It contains among other things the files of the Taxonomy Helper, where also getParent is inside. Thus this Main file supplies the results for the generalization. This file is among other things also responsible for the fact that the generalization can be combined with the abstraction.

2.3 Taxonomy Helper

The class 2.3 Taxonomy Helper is called in all main files and helps to access the imported taxonomic structure of the ontologies. It can be used in any other operator by importing it.

2.4 Ontology Helper

The Ontology Helper class accesses the ontologies of the two domains and is imported into the main files. The file is in the same repository as the Taxonomy Helper and can also be re-imported into different classes.

2.5 Abstraktionsoperatoren

A total of five abstraction operators were defined. These are described conceptually in the master's thesis by Xinyuan Cai. All abstraction operators are modularized in a separate Java file and form a subclass of the Java file "Operators".

2.6 Generalisierungsoperatoren

Parallel to the abstraction operators, the generalization operator was implemented. This can also be combined with the abstraction operator. In addition, a bound can be used with the operator, which makes it possible to generalize over several hierarchy levels.

2.7 GreedyTransferProcess

This class combines the results of Xinyuan Cai's work with that of Nils Gormsen. The tasks are transferred to the target domain using the analog mapping and abstraction.

2.8 Commpund Abstraction

The file contains the results of the workflows to which all abstraction operators were applied simultaneously.

2.9 Data Sets

The database for both domains is located in the EVER 2 data set in the Evaluation superfolder. A separation is made between the two domains. Both domains contain all workflows and their task hierarchies. The EMISA 2015 data set, which can also be found there, is only used for testing purposes and is not relevant any further.

3. Tool-Chain

Begrifflichkeiten:

Graphviz-Java (installed over API): Graphviz is open source graph visualization software. Graph visualization is a way of representing structural information as diagrams of abstract graphs and networks. It has important applications in networking, bioinformatics, software engineering, database and web design, machine learning, and in visual interfaces for other technical domains.

The Graphviz layout programs take descriptions of graphs in a simple text language, and make diagrams in useful formats, such as images and SVG for web pages; PDF or Postscript for inclusion in other documents; or display in an interactive graph browser. Graphviz has many useful features for concrete diagrams, such as options for colors, fonts, tabular node layouts, line styles, hyperlinks, and custom shapes. (Quelle: https://graphviz.org/)

4. Next Steps

The next step of the EVER II project is to try to implement the golden standard of ontology mapping into the code. A similar procedure will be chosen as for the analog mapping. In addition to this, the cost function has to be integrated to allow the best possible transfer. The project is always evolving, so this file is also always changing. Changes should always be well documented, so that new project participants can easily familiarize themselves with them.