A customizable importer for the clinical data warehouses PaDaWaN and I2B2

Georg FETTEa,b,[[1]](#footnote-1), Mathias KASPARb, Georg DIETRICHa, Maximilian ERTLb, Jonathan KREBSa, Stefan STOERKb, Frank PUPPEa

a Würzburg University, Chair of Computer Science 6

b University Hospital of Würzburg, Comprehensive Heart Failure Center

**Abstract.** In recent years, clinical data warehouses (CDW) storing routine patient data have become more and more popular to support scientific work in the medical domain. Although CDW systems provide interfaces to import new data, these interfaces have to be used by processing tools that are often not included in the systems themselves. In order to establish an extraction-transformation-load (ETL) workflow, already existing components have to be taken or new components have to be developed to perform the load part of the ETL. We present a customizable importer for the two CDW systems PaDaWaN and I2B2, which is able to import the most common import formats (plain text, CSV and XML files). In order to be run, the importer only needs a configuration file with the user credentials for the target CDW and a list of XML import configuration files, which determine how already exported data is indented to be imported. The importer is provided as a Java program, which has no further software requirements.

**Keywords.** data warehouse, ETL

# Introduction

PaDaWaN [1] and I2B2 [2] are clinical data warehouse (CDW) systems, which store their data in an entity attribute value (EAV) model [3]. Both systems can be loaded via a web interface or data can be directly written into the respective database tables. However, further tools are required that read data from a data source (and optionally transform the data as desired) and send it to an import interface in order to get the data into the CDW.

The goal is to provide an import tool for PaDaWaN as well as I2B2, which is able to import the most common import data formats: plain text for unstructured data, e.g. discharge letters and reports; CSV for structured data, e.g. coded diagnoses or laboratory values; XML for semi structured data, like PMDs (see below), which are the majority of document types at university hospital of Würzburg. The import tool should be easy to configure, easy to integrate into arbitrary ETL workflows, and should not require additional software.

# State of the art

IDRT [4] contains an import and mapping tool that can be used to import catalog and fact data into I2B2. The IDRT tool can be either used as a stand-alone desktop application or it can be used as a component in the ETL suite Talend Open Studio[[2]](#footnote-2). When IDRT is to be integrated in other ETL environments this is not yet possible out of the box. Also IDRT is not able to process arbitrary XML import files. As the IDRT tool is only able to import into the I2B2 data model we chose to create a new importer tool that has a more abstract data model interface and is easier to integrate into arbitrary ETL tools.

# Methods

The abstract data model of PaDaWaN, which is quite similar to I2B2, is depicted in Figure 1. It consists of a catalog of attributes (concepts; in I2B2 terminology) and a large set of facts (observation facts). Each fact is linked to a catalog attribute (concept) and is attached to a mandatory patient (patient) and, optionally a patient case (encounter) and/or a reference (instance).

**Figure 1:** Abstract data model of PaDaWaN and I2B2

Due to the similar data models, a common Java interface for both PaDaWaN and I2B2 was created. The interface unites the shared properties of both systems, currently still neglecting some specific details of I2B2 [5] (e.g. PaDaWaN stores only the facts’ measurement timestamps, whereas I2B2 supports start-end-intervals). Using this interface adapter implementations for PaDaWaN, as well as for I2B2 were created. The importer can be configured into which target system to import into. The importer is realized as a set of configurable Java classes for each data format (plain text, CSV, XML), which are altogether provided as a .jar-file. In order to be run, the program needs a configuration file specifying the import target system (CDW type, Server URL, database name) and the corresponding user login credentials. Furthermore, the importer needs the path to a file system directory containing import configurations (ICs), which define what data and how this data has to be imported. Both import parameters are provided to the .jar-file when started via command line parameters.

An IC is an XML file, providing all necessary information for the import process for one data source. An IC specifies a file system directory from which the importer imports all contained files. All further configuration is dependent on the data type of the exported data:

**Plain text files**: The text content of each file from a given folder is imported as a single textual fact. The fact’s metadata (e.g. patient ID) has to be extracted from the file’s filename. The IC has to include regular expressions, which provide the identification of the needed metadata within the filename (e.g. Figure 2).

**Tabular files**: All files in a given folder are comma-separated value (CSV) files, which all have to have the same format. The IC configuration has to contain the syntactical format specification (e.g. delimiter, encoding, escape characters, etc.) and the semantic format specification. Via the tabular files importer the following data can be imported:

* ***Catalog data***: The catalog for a specific CDW domain can be imported from a tabular file. The IC has to determine which columns in the CSV file shall be interpreted as the name, the catalog attribute ID, the data type, the unit or further information that is needed for the creation of catalog entries.
* ***Fact data***: For the import of fact data from a CSV file the IC has to specify which columns contain the catalog attribute ID, values, measuring time stamps of each fact, etc. (e.g. Figure 2).
* ***Import-metadata***: To ease the import of some data sources, patient-, case- and document-metadata can be imported into dedicated import-metadata database tables. These tables can be used when the importer is importing data lacking necessary information, e.g. only case IDs are provided instead of patient IDs. The importer can obtain the corresponding patient IDs from the import-metadata. Another example is in case the source data only contains fact measurement time stamps and no corresponding case IDs. The lacking case ID for a given fact measurement timestamp can be calculated by taking the temporally nearest case from the import-metadata tables.
* ***Age***: The stored age for each case is the patient’s age at the beginning of the respective case. For this calculation, the patient’s date of birth from the import-metadata tables is used.
* ***Authorization data***: PaDaWaN possesses an authorization concept managing users, groups, memberships of users to groups and accessibilities to catalog attributes for each group. All information belonging to those domains can be also be imported via the tabular files importer.

**XML files:** All files in a given folder have to be XML files with the same XML schema. The IC specifies a list of tuples (XML tag, XML attribute1, XML attribute2), which are searched for when the importer traverses the XML during import. For each tag, which is contained in the list of tuples, a fact is created with the catalog attribute identifier given by the XML attribute1 and the fact value given by the XML attribute2. Additional fact metadata (e.g. measure timestamp, patient identifier, etc.) have to be extracted from the filename of the XML file, as it is done for plain text files.

|  |
| --- |
| <ImportConfig><DataCSV Project="Diagnose" Dir="DiagExport" AttrIDColumn="DKEY1" ValueColumn="DITXT" CaseIDColumn="PFALNR" MeasureTimestampColumn="DIADTZT" MeasureTimestampFormat="yyyyMMddhhmmss" Encoding="UTF8" /> <DataText Project="Arztbriefe" Dir="BriefExport" AttrID="Brieftext" DocIDRegex="^(\d+)\_.\*" Encoding="UTF8" /></ImportConfig> |

**Figure 2:** Example of an import configuration including a CSV data source and a plain text data source

When run, the importer logs potential errors or warnings that occurred during the import process into a designated table of the target database. After a successful import of an import file, this success is also logged, storing the filename and the file modification timestamp of the respective file. When repeatedly running the importer with the same ICs on the same data source folders the importer can be configured to use the database logs to prevent unnecessary processing of already processed files. Using this mode, the import folders can be re-used for iterative (e.g. daily) export of data deltas without the need for cleaning those folders before every update cycle.

# Implementation

The presented customizable importer is currently deployed and in use in the CDW projects at the university hospitals of Würzburg and Ulm. Within the CDW project Würzburg the majority of data domains that are loaded into the CDW are processed using the presented customizable importer. The domains processed by the importer at the current stage of the project are (grouped by data format) **CSV**: laboratory reports, diagnoses (ICD10), disease-related groups (DRG), procedures (OPS), age, sex, admission and discharge time stamp, hospital ward; **XML**: echocardiography reports, sonography reports, anamnesis and physical examination reports, electrocardiogram reports, coronary angiography reports, X-ray reports; **Plain text**: discharge letters.

A small fraction of the data domains (e.g. medication) still require additional transformations, therefore their import is performed by Java programs, which are individually created for those data domain.

At the CDW project in Ulm, which is currently still in the prototype phase, the importer processes the domains: **CSV**: diagnoses, laboratory reports, age, sex, admission and discharge time; **Plain text**: discharge letters.

# Lessons learned

The use of the customizable importer improved the rollout speed at the Würzburg CDW project for the integration of new domains into the CDW, especially for the XML data domains. The XML data domains originate from so-called Parametrized Medical Documents (PMD), which are exported from the local SAP system in a generic XML format. PMDs are hierarchical structured documents comparable to CDAs (Clinical Document Architecture) [6]. At the university hospital of Würzburg, there exist hundreds of PMD domains [7], which represent a big fraction of the total data stock in the local hospital information system. The import process for those domains can now be configured via ICs instead of having to write Java code. Although in both CDW projects where the importer is used (Würzburg, Ulm) the technicians configuring the ICs are also programmers, the configuration of ICs is preferred to writing Java code. The training period for new workers in the ETL process becomes shorter because no programming IDEs have to be installed and configured and no code framework has to be learned. As part of the future development, the importer could be equipped with a user interface and an IC generation wizard like it is the case in IDRT [4] or in the CSV import wizard of Microsoft Excel.

By using the customizable importer and its import configurations the import process is logically separated from processes belonging to another technological level (e.g. database transactions, file access, program logic, etc.). This separation improves the documentability of the loading part of the ETL process. Other people in the CDW project, e.g. working on quality assurance, can more easily review the import process because all import logic is concentrated in the single folder with IC-files.

The presented importer serves solely for the loading of data and not for any transformations of processed data. For transformations, an additional ETL software has to be used. In the CDW project at the university hospital of Würzburg the importer is included in an additional Java program that controls the ETL process [1]. As the customizable importer only needs text file configurations, an integration in another ETL environment should be easy to accomplish.

The abstraction from the concrete target CDW keeps the importer independent from the CDW system used. When choosing to exchange the underlying software (e.g. switching from I2B2 to PaDaWaN or vice versa) the existing ICs can be re-used without any further adjustments. Other CDW systems could be added to the framework by creating an appropriate adapter implementation for the abstract model (assuming the data model matches the abstract model depicted above). Due to the similar EAV models of PaDaWaN and I2B2 the unification under the same interface did not pose major problems. Other possible CDW data models like FHIR QI-Core[[3]](#footnote-3), OMOP CMD[[4]](#footnote-4) or PCORnet CDM[[5]](#footnote-5) are more complex and would need a more sophisticated abstract model before a unification under a common interface could be achieved.

# Conclusion

A customizable importer for the two DW systems PaDaWaN and I2B2 was presented which is able to import data in the most common data formats (plain text, CSV and XML). In order to run the importer only needs a configuration file with the credentials of the target DW system and a list of XML import configurations, thus making the importer easy to integrate into ETL workflows.

# Conflict of Interest

No conflicts of Interest

References

1. G. Fette, M. Ertl, G. Dietrich, et al., An improved data workflow for a medical data warehouse, *European Journal of Epidemiology* **32** (2016), Health-exploring complexity: an interdisciplinary systems approach HEC2016, 54.
2. S. Murphy, G. Weber, M. Mendis, et al., Serving the Enterprise and beyond with Informatics for Integrating Biology and the Bedside (i2b2), *J Am Med Inform Assoc*. **17 (2)** (2010), 124-130.
3. V. Dinu, P. Nadkarni, Guidelines for the effective use of entity-attribute-value modeling for biomedical databases, *Int J Med Inform.* **76 (11-12)** (2007), 769-779.
4. C. Bauer, T. Ganslandt, B. Baum, et al., The Integrated Data Repository Toolkit (IDRT): accelerating translational research infrastructures, *Journal of Clinical Bioinformatics*. **5 (Suppl. 1)** (2015), S6.
5. L. Liman, Vergleich und Integration der Data Warehouse Architekturen PaDaWaN und i2b2, *Master thesis (german)* (2017)
6. R. Dolin, L. Alschuler, C. Beebe, et al., The HL7 Clinical Document Architecture, *J Am Med Inform Assoc*. **8 (6)** (2001), 552-569.
7. M. Ertl, Erfassung von klinischen Untersuchungsdaten und Transfer in ein Data Warehouse, *Master thesis (german)* (2011).
1. Corresponding author: Georg Fette, University Hospital of Würzburg, DZHI, Am Schwarzenberg 15, 97078 Würzburg, Germany; E-Mail: georg.fette@uni-wuerzburg.de [↑](#footnote-ref-1)
2. https://www.talend.com/products/talend-open-studio/ [↑](#footnote-ref-2)
3. http://hl7.org/fhir/us/qicore [↑](#footnote-ref-3)
4. https://www.ohdsi.org/data-standardization/the-common-data-model/ [↑](#footnote-ref-4)
5. http://www.pcornet.org/pcornet-common-data-model/ [↑](#footnote-ref-5)