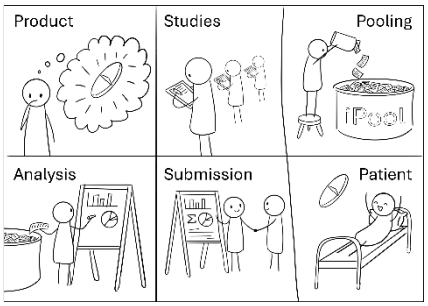
Paper DH01

A Splash in Time – Insights into a Data Pooling Time Machine

Lutz Hoffmann, Bayer AG, Berlin, Germany
Dana Vinzelberg, Bayer AG, Berlin, Germany

ABSTRACT
A SAS® macro system (iPool) was designed to support integration of clinical study data into data pools in a highly efficient manner. As a pool (metadata and data) changes over time like a "living object" all necessary modifications can be maintained as pool definitions are stored separately from the data. The data itself is stored only once in a pool. In addition, medical coding (e.g. MedDRA and WHODD) is supported for multiple versions. A new pool is initially defined with pool metadata and controlled terminology. Afterwards studies can be added, and pool transformations can be defined with the according timeframe information. Therefore, it is possible to extract the pooled study data as it was at a certain point in time in the pool's lifetime. With this approach it is possible to "walk back in time" to extract a pool from the past.

INTRODUCTION
iPool is a tool to maintain integrated clinical data for analysis, aka data pool. It is a definition-driven tool, which means that all operations performed in iPool are controlled by the definition datasets. All operations are performed by calling the according iPool-macro.
The task is to integrate the data from different studies into one pooled area and provide this pool data to the consumer. This is a central point in the clinical development on the way which begins with a substance and ends at the patient.



Due to the fact, that the existing data integration environment will be retired, the idea for a new tool in a new environment was born. To overcome certain limitations, e.g. the embedding into a proprietary platform, a pure SAS® solution was requested. One focus in the concept is to also handle big datasets in an *efficient* and *flexible* manner, another aspect is to ensure data *integrity*. The data integrity is achieved by checking the consistency of the metadata definition and the study data and finally the pool data.

CONCEPT
The basic design concept is to define the different aspects of a pool in so called definition data. This defines the data structure and the updates on data. As the pool is a living object which changes over time this aspect is fundamental in all definition datasets. By keeping the full history of the pool definitions, it is possible to walk back in time and select the pool as it was for a specific point in time in the past.

ARCHITECTURE
The core elements for a pool are

- metadata and codelists definition
- study and study-data registry
- study-data and -metadata
- data modification definitions

See also "Elements" below.

WORKFLOW

The general workflow for maintaining a pool is

- Create pool area
- Define metadata and codelists
- Register study
- Load study data
- Define modifications on the data
- Extract of the pool (data, metadata and codelists)

LIFECYLE

iPool fully supports the development-validation-production (dev-val-prod) lifecycle approach.

DESIGN

Three design aspects are crucial for iPool:

- Effectiveness
- Flexibility
- Data Integrity

The effectiveness aspect is achieved by minimizing space consumption and using hash-objects whenever useful. With allowing many changes, e.g. on metadata, codelists, and data, a high degree of flexibility is given.
To ensure the data integrity there are many checks on metadata and codelists against study data and finally the current pool data when loading new data for a study into the pool. Also, all changes of definitions, e.g. metadata or change of the data, are checked for inner consistency and finally the resulting pool data is checked.

MINIMIZING SPACE CONSUMPTION

The four principles of data storage in iPool are

- store study data only once “as is” and never delete it
- allow new versions of study data
- allow extension of study data (add records)
- provide pool data as views instead of datasets

CHANGE OF DEFINITIONS

All definitions can be changed by exporting the definition, updating and loading the definition back into iPool. All changes can be loaded simultaneously. When loading these definitions, the consistency checks for changed definition and for the resulting pool are performed to guarantee the pool integrity.

CONSISTENCY CHECKS

There are multiple checks performed covering the aspects of

- Metadata & Codelists vs. Study Data (during load)
- Metadata & Codelists vs. Pool Data (during load of study data and/or pool definitions)
- Changes in Definitions

The central aspect of checking metadata & codelists vs. study or pool data is that certain attributes from the metadata and code and decode values from the codelists are checked against the data. These are structural checks independent from the data model, e.g. SDTM or ADAM, and don’t cover other checks like those in Pinnacle 21®.
There are checks on pool level to ensure that at least one common version for MEDDRA and WHODD is available and only one code/decode for the same meaning.

RETRIEVE POOL ON DEMAND

When finally retrieving the data from the pool the data is provided via views. All defined changes on the data are applied to the stacked study data in the views. In conjunction with the pool data the according metadata and codelists are extracted.

ELEMENTS

The basic types of pool elements are definition-, registry-, and data datasets. As definition datasets can be modified, the registry datasets are updated by iPool processes. In the end, both types of datasets (definition- and registry-datasets) define the pool, while data datasets are the stored study data and, in combination, results in the pool data base.

METADATA AND CODELISTS

The basic components to define the data structure of the pool data is achieved with metadata and codelists.

As codelists are maintained by a standards department, most of the codelists used are predefined. These can be integrated into the pool by defining the locations of the codelist areas. In addition, pool specific codelists can be defined.

The definition of metadata is done in two datasets: metadata and meta-metadata (which defines the structure of the metadata).

There is a base definition of iPool metadata with common data attributes, e.g. column name, type, length, label, codelist and other attributes which have a specific meaning in iPool only. As metadata are also crucial for consuming tools, the definition of the structure of metadata can be defined via meta-metadata. This allows to create and maintain all necessary attributes needed beyond iPool.

STUDY REGISTRY

To load study data into the pool the *study* must be registered in iPool beforehand. This registration is stored in the study-registry.

STUDY DATA, STUDY METADATA, AND STUDY DATA REGISTRY

All study datasets are stored as a simple copy with an internal name. This storage concept allows new versions of the data, if necessary, by storing the according information in the *study-data-registry*. The data in the study data datasets are always stored “as is”. There are no modifications performed on the stored data over time. While loading the study data, also the metadata of these datasets is stored in the study-metadata.

MODIFY DATA DEFINITIONS

Modification of data is supported by certain types of changes.
iPool supports the change of specific values in the data, the removal and the addition of records.
To add records to an existing dataset a dataset containing these new records can be loaded as an “add” dataset.
Beside these, two specific change definitions are supported: UpMap of code and decode columns and Glossary Coding Update.

DATA-CHANGES

In Data-Changes the change of dedicated values in specific columns can be defined.

REMOVE- RECORDS

In Remove-Records the removal of specific records can be defined. The reverse operation (add records) is solved in a different way.

UPMAP

UpMap is a very specific technique to harmonize the data based on code and/or decode changes in codelists. As columns can be linked as code-decode pairs a change in the codelist effecting code and/or decode these changes must be performed in sync. This is supported by the UpMap operation. Due to some extra information in the codelists that an UpMap (newer) code/decode is available for an (outdated) code, the outdated code/decode must be replaced (updated) to the newer one which is used in at least one (new) study. Because this use of different codes with the same meaning is checked during the load of new study data the pool must be updated accordingly because the load of inconsistent data is refused.

GLOSSARY CODING UPDATE

iPool supports updates to new versions of MEDDRA and WHODD in the (so called glossary coding) update. By design iPool requires to have at least one common version of MEDDRA and WHODD for all studies and affected datasets in the pool. In case of loading study data with a newer version than the latest (common) in the pool, the pool must be updated to this newer version to load the data.
If multiple versions exist, the user can choose during the extract of data which one to use. By default, the latest version is chosen for extract.

POOL DATA VIEWS

Views for pool data are created for extracting the data from the pool. With a defined data structure and all study datasets linked into the basic stacking of data is given. All changes are applied via statements which implement the change directly in the view. Using static paths to the study data instead of libnames ensures the availability of the data without the need to have some specific initialization steps.

WORKFLOW I – INITIAL STEPS

To start working on a pool the pool area must be created, i.e. all necessary folders are created. After that the pool metadata and codelist must be defined. When this is finished the study to be loaded must be registered and then the study data can be loaded into the pool. This can be repeated for all studies needed in the pool. In parallel modification of data can be defined. Finally, the pool data can be extracted to allow the analysis of the pool data.

CREATE POOL AREA

As the environment is based on the lifecycle “dev”, “val”, and “prod”, all these stages must be created in the pool area. When creating such a stage all definitions are stored (empty) in the definitions folder plus one dataset containing the pool core information which includes the underlying data-model, like SDTM, ADAM or LEGACY. Over time these definitions will be updated without removing any information but adding new or changed definitions and deactivation of outdated parts.
Not removing any outdated information from definitions is crucial to walk back in time and make iPool a data pooling time machine.

CODELISTS AND METADATA

The load of *codelists* is possible in several ways. The most common way is to pick standard codelist from specific locations. For this the paths are stored in the codelist-locations dataset. All codelists datasets found in these paths are loaded into the pool. A second option is to store codelist datasets in a folder “codelist” in the pool area. The third option is to store the codelist information into a codelist-definition dataset and load this into the pool.

Having codelists loaded into the pool the definition of the metadata can begin. In case the base *metadata* structure needs to be extended this can be done be updating the *meta-metadata* to e.g. add more columns or extend e.g. the length of existing columns in the metadata. With this the *metadata* provided in the pool extract can be extended to the needs of the consuming analysis system.

Based on the *meta-metadata* the *metadata* must be provided in the according data structure. In the metadata the data structure of all pool datasets is defined. There are certain iPool specific attributes in the *metadata* which are used to perform operations like UpMap or Glossary Coding Update.

REGISTER STUDY

Before loading study data into the pool, the study must be registered. The registration of a study is one of the rare operations which cannot be reverted.

LOAD STUDY DATA

After the registration of a study the study data can be loaded into the pool. Only datasets which are defined in the metadata can be added into the pool. All datasets added must have exactly the data structure as defined in the metadata. Values in columns where a codelist is assigned (either code or decode) must have only values which are defined in the according codelist. More checks like NOTNULL or key-columns have unique values are performed according to the metadata.

EXTRACT

After the first load of study data the pool can be extracted as needed. Later in time the extract allows to walk back in time and extract a historical version of the pool. All studies and datasets available can be chosen. By allowing to choose a MEDDRA and WHODD version - depending on availability - there is an additional degree of flexibility for this.

WORKFLOW II – POOL UPDATE

Beside adding more studies and study data into the pool, also almost all other aspects of the pool can be changed over time, like loading updated study data, modify metadata or define data changes.

LOAD OF UPDATED DATA FOR EXISTING STUDIES

Whenever a dataset needs to be replaced with a newer version, this can be done easily by adding the modified dataset again. This will create a new dataset in the pool and the new dataset will be used instead of the old one from the timepoint of loading. Nevertheless, the old dataset remains in the pool for two reasons.

1) The previously created views rely on the existence of this dataset.
2) To allow walk back in time before timepoint of loading a new version of the data, this “outdated” data must be still assessable.

MAINTAINING DEFINITIONS

In general, the internal iPool definition files have an external pendant which must be used for modifying the definition. The external definition contains the current status of the internal definition. To apply the changes this external dataset must be updated and then loaded into iPool. This load stores the updated definition in the internal definition dataset.

METADATA MAINTENANCE

There are several modifications allowed in meta-metadata to allow new columns in the metadata or update of existing ones. Add new rows to meta-metadata to add new columns to the metadata, modify certain attributes of the metadata column definition. Some changes are revoked, e.g. it is not allowed to shorten the length of a character variable or change the type from numeric to character and vice versa. As iPool uses several columns of the metadata for processing, the definition of these columns is fixed.

More modifications are allowed in metadata, as the data structure of the pool must not be limited. The only restrictions on changing attributes are that it is not allowed to shorten the length of a character variable or change the type from numeric to character and vice versa or to change the values in the ID-columns, which are handled internally by iPool.
To modify the structure of the pool data it is allowed to add, rename, or delete columns and datasets. For adding new columns or a dataset, new records must be added to the metadata. To rename a column or dataset the according name column (column-name or dataset-name) must be changed. And to delete a column or dataset the value in the according name column must be removed.

HARMONIZATION

Sometimes there is the need to modify existing data in the pool to harmonize code/decode usage when loading a new study, e.g. in pool an outdated code is used but in the new study the current code is used. The according UpMap of the pool to change the outdated code to the currently used in the new study can be performed in the same maintenance step as the load of the new study data.

Similar constellations may occur for MEDDRA and/or WHODD, e.g. in pool the latest available MEDDRA version is 27.1 but the new study is coded with MEDDRA version 28.0. In this case, the pool must be updated to MEDDRA version 28.0 or new study must be “updated” to version 27.1. Even better would be to do both updates (pool MEDDRA version to 28.0 and new study to 27.1) to allow more flexibility for the analysis, because the user than can choose either 27.1 or 28.0 when extracting the data from the pool.

MODIFY DATA

Several operations to modify the data in the extracted views are supported by iPool: data-change, add-records, remove-records, UpMap of code and decode columns, and Glossary-Coding-Update (GC-Update). Four of these operation types are implemented via definition datasets. The only modify data operation which works differently is “add records”.
To add one or more records to an existing dataset a dataset with the records to be added can be defined and loaded in “add records” mode. By this the “base” dataset is appended by the “add” dataset. This operation can be repeated.

DATA-CHANGE

To change a specific data point the data-change operation can be used. With this, corrections in the data can be done, e.g. when receiving an errata list which describes data issue detected after study database lock.

ADD-RECORDS

To avoid the re-load of an existing dataset for just adding some records iPool support the add-records functionality. With this a “add” dataset to the “base” dataset is loaded which is used in the extract (view creation) as an extension of the base dataset. “Add” datasets are valid until the base dataset is “replaced” by a newer version.

REMOVE-RECORDS

The removal of records can be defined in the according maintenance dataset. Records to be removed must be identified by a specific condition which uniquely identifies these records.

UPMAP

The UpMap operation is a specialized Data-Change operation. It allows the change of so-called code- and decode-variables in one step. As the change is linked to codelists and codelists can be assigned to multiple columns in multiple datasets one definition of change can be applied to multiple columns in multiple datasets.

GLOSSARY-CODING-UPDATE

The update of MEDDRA and WHODD is the most complex data modification functionality. To perform the update specific update datasets are merged in a defined way and specific columns are updated. As the MEDDRA and WHODD dictionaries are also used in this, this is a “two-level merge”. In addition to these two updates a simple join functionality in conjunction to MEDDRA and WHODD merge is provided to allow, e.g. join of Drug Grouping information in conjunction with the update to a specific glossary coding version.

INTEGRITY

All operations on the pool are checked to ensure the pool integrity. If any inconsistency is detected the maintenance operations are aborted and an error message and report is created.
The execution of almost all checks is controlled by so called check-suites, which contains a list of checks which are performed one by one. All check results are then collected into a consolidated check result dataset for this particular check-suite, which is the base for the check report, which is created in case of check violations. If any check is violated the program stops before storing anything into the pool.

POOL CONSISTENCY CHECK

All maintenance operations on the pool finally extract the *current* pool data and these are checked for consistency (aka pool consistency check). With this it is not possible to load any data or definition which may corrupt the pool. As the pool data growth over time, this check part is the most time-consuming part in the end.

CHANGING DEFINITIONS

When loading a definition dataset into the pool, this is first checked for structure, values and allowed changes before proceeding the final pool consistency check.

LOADING STUDY DATA

When study data is loaded into the pool, the data is checked against metadata and codelists, including evaluation of e.g. --CODLST column against corresponding code and decode columns before proceeding the final pool consistency check. This ensures reduced run-time if already an issue in the study data is detected before the final pool consistency check.

TIME

As a pool changes over time, it is regarded as a living object. These changes occur on pool structure level as well as on data level. With storing appropriate time information in all definition- and registry-datasets the state of a pool at a certain point in time can be reconstructed.

Based on the fact that the data, once stored in the pool, stays as is and all changes on the data are achieved by storing modification-definitions, which are applied during the extract, the walk back in time can be done without complex calculations or queries. Therefore, the extract of pool data, metadata, and codelists can be done easily, because data and modifications are retrieved and applied for the point in time as requested.

For the extract of the pool data the user can specify a point in time between the first data load and now. By specifying a timepoint in the past the pool definitions are regarded as they were at this point time. Only those studies, datasets and data modifications, including glossary coding thesauri versions, which were available at this point in time are applied. This allows the exact reconstruction of the pool data, metadata, and codelists as they existed at a specific timepoint in the past.

Unfortunately, iPool as a time machine is not yet supporting is a look into the future. If we achieve this, we will rename it to crystal ball ;-)

CHALLENGES

One of the biggest challenges, besides getting enough resources for tool development, was the design of the glossary-coding update. The tricky thing is, that the specific version of a thesaurus must be applied for all studies and all datasets coded with this thesaurus (in our case MEDDRA, which is applied to several domains) to have this as a common version - pool wide over all studies. Adding a new dataset to the pool which is also coded with this thesaurus requires that the dataset is already coded with a common pool version or needs the according GC-Update when loading the data to ensure that at least one common version for this thesaurus exists in the pool.

Another challenge was the introduction of a pool-tag, which allows modification definitions not only for a specific study but for the whole pool. The tricky thing is, that this pool-tag must be linked to the timepoint of definition, because if you add a study later on, the modification definition does not necessarily apply to this new study as well. Otherwise, you really need a functioning crystal ball.

As we spent quite some time on the design of the iPool metadata, allowing changes to this metadata were not a big challenge, may be also because we defined a lot of rules for allowed changes which are checked by program.

CONCLUSION

With the approach of storing the data only once “as is” and combining the data and data modifications in views the space consumption is minimized. By allowing the selection of different timepoints or glossary coding versions when extracting the data a high flexibility can be ensured – which require some effort on maintaining these glossary coding versions, of course.

From the first day on the pool changes over time.
This fact is a central aspect in the design of iPool and the architecture of all operations on metadata, codelists and data are taking the current time into account. Once stored definitions are never deleted but get a valid time frame information. With this, every definition can be selected for a specific timepoint and used for processing. As the loaded data is registered with same approach of valid time frame information, all aspects of the pool can be controlled over time.

To evaluate the design aspects, we perform a couple of test and compare results with the experiences when using the existing data integration system. This is not to blame the existing system, because this was build a long time ago and was not updated for a very long time. And we were able to learn from the pain points using this which also leads us in the design phase of the iPool project. We therefore also spent some time to redesign and reengineer the consistency checks of the old system.

Here are some results from the comparison between current system and iPool.

EFFICIENCY

Storage space required is minimized as study data is stored only once. No audit trail or copies of datasets are required.
Reduction of runtime. Pool maintenance tasks run 5 to 10 times faster.
Performing multiple changes to the pool at the same time also reduces runtime drastically.

FLEXIBILITY

Flexible definition of metadata.
Higher flexibility because multiple MEDDRA and WHODD versions can be made available.

INTEGRITY

Because more checks on the pool data integrity are performed, e.g. a check of --CODLST against CODE and/or DECODE of given codelist in --CODLST was not available in the old system, the quality of the provided data is enhanced.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Author Name: Lutz Hoffmann
Company: Bayer AG
Address: Müllerstr. 178, 13353 Berlin
Work Phone: +49.173.6747038
Email: lutz.hoffmann@bayer.com
Website: https://www.bayer.com

Co-Author Name: Dana Vinzelberg
Company: Bayer AG
Address: Müllerstr. 178, 13353 Berlin
Work Phone: +49.30.221540214
Email: dana.vinzelberg@bayer.com
Website: https://www.bayer.com