Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 00 (2016) 000–000

Procedia Computer Science

www.elsevier.com/locate/procedia

13th International Conference on Current Research Information Systems, CRIS2016, 9-11 June 2016, Scotland, UK

Data quality assessment and improvement: a Vrije Universiteit Brussel case study

Steven Van den Berghe*, Kyle Van Gaeveren

Vrije Universiteit Brussel, R&D Department, Pleinlaan 2, 1050 Brussel, Belgium

Abstract

As the Vrije Universiteit Brussel switched from an in-house built CRIS to Pure, a large number of data quality issues were discovered. In order to solve these, a large-scale data quality assessment and improvement program was started. The assessment sought to find data quality issues and prioritize cleaning tasks along different dimensions, such as reusability and complexity, while taking into account compliance and stakeholder happiness. Moreover, in doing these assessments, an attempt was made to isolate relatively easy to clean parts of the data in order to make them more feasible for people with less domain-knowledge. Finally, some of these data quality improvement operations turned out to be straightforward enough to fully automate them.

Peer-review under responsibility of the Organizing Committee of CRIS2016.

Keywords: Data Quality Assessment; Data Cleaning; Data Quality Improvement; Research Data Management

1. Background: Research Information Management at the Vrije Universiteit Brussel

The Vrije Universiteit Brussel (VUB henceforth) has been keeping track of research output, projects, activities and their associated metadata since the 1980's. For the most part, this information was stored in a custom-built application, R&D-net, which went to a number of iterations over time. The CRIS at the VUB is not only used to store information on research output and activities, but is also used to pass on financial project information to the financial administration. While R&D-net was used for a long time, numerous complaints were voiced by users, both

* Corresponding author. Tel.: +32 497-48-21-40.
E-mail address: svdbergh@vub.ac.be

Peer-review under responsibility of the Organizing Committee of CRIS2016.

on the research and on the administration side: performance was sub-optimal, integration with external systems turned out to be problematic and by adding features over the years, the templates had become cluttered with ambiguous fields. Moreover, the user interface was not user-friendly and there was a distinct lack of administrator's tools to facilitate

In 2012, the decision was made to migrate to a new, commercial off-the-shelf application, Pure. There were numerous reasons for moving away from in-house development to a commercial system, the main ones being an increased need for interoperability with external systems, for improvements in performance and user-friendliness and a desire for decent tooling to deal with data quality issues. While it would be possible to integrate these requirements in an in-house developed system, it was decided that purchasing Pure would help us overcome the significant technical hurdles we were faced with.

In this paper, we want to share some of our experiences of migrating from the old system to the new one, with particular emphasis on matter of data quality. While our experiences are definitely not unique, we are convinced that some of the methods and techniques we employed can be useful for other organizations who are faced with a similar data quality problem.

1.1. Migrating from R&D-net to Pure

The migration from R&D-net to Pure proved to be the toughest hurdle in the entire implementation project. In order to deal with the legacy data, the implementation team was faced with the choice between three options: the first option was to start afresh and leave the old data behind. While this had the advantage of not having to deal with the intricacies of a large-scale data migration, it would also mean that all historic data would be lost. Therefore, this option was quickly dismissed. A second possible approach was to clean the data in the legacy system and then migrate. While this seems like the obvious way to handle this, there were a number of factors that made this impossible or impractical. R&D-net provided very little tools to allow administrators to detect and resolve data quality issues. Moreover, since some of these issues were a consequence of the way R&D-net had structurally evolved over the years, any data cleaning effort itself would be hard to accomplish. The final approach, which is the one we chose in the end, was a combined effort of minimal cleaning pre-migration and setting up a large-scale cleaning effort post-migration. Drafting rules to migrate the data was a complex endeavor, but in the end, a nearly complete migration was achieved. Nevertheless, most of the data quality issues that plagued the legacy system were simply transferred into the new system during the migration, necessitating the need for a large-scale data quality assessment and improvement exercise.

1.2. Introducing the new CRIS to users and administrators

Overall, the new system was met with positive criticism by researchers and most of the administration. However, as Pure provided a much more transparent view on the existing data, one of the most-heard complaints was that the data quality had been diminished. While this was objectively not the case (nowhere had 'correct' data been transformed in 'corrupt' data), from the researchers' point of view, this objection was understandable: the new system provided more functionality to actually use the data (e.g. in CVs and reports) and provided them with greater opportunities to view other researchers' data in a much more accessible interface, which made existing data quality issues more visible. While the core of such functionality was already present in R&D-net, the tooling was not sufficiently mature to encourage wide-spread use. Administrators were also provided with greater access to the data in different ways (reports and dashboards) and were also exposed to data quality issues. Moreover, throughout the implementation phase, relatively little attention was given to communication with the broader user community (user acceptance testing was limited to a small group that remained unchanged throughout the implementation), which led to some researchers and administrators assuming that the go-live would somehow solve all existing data quality problems. Of course, since the actual cleaning effort had yet to start, this led to disappointment in some cases.

In order to 'evangelize' the new system, a total of 40 demo sessions were organized, both at the Etterbeek and Jette campuses. These sessions turned out to be highly valuable, especially after a few months, since many researchers would use them to give feedback about their experiences with the system. While some of their questions

could be attributed to the learning curve associated with the introduction of a new system, others clearly showed us data quality issues we were not aware of at the time. It also allowed us to encourage users to clean up their own data to a certain extent and to popularize a program in which we would support such data quality improvement projects (see 2.2).

2. Data quality assessment and improvement

The literature on data quality assessment and improvement recognizes three distinct phases in a data quality methodology: 1) contextualization, 2) assessment/measurement and 3) improvement[1]. In this paper, we will primarily be focusing on the latter two phases, as the contextualization of many data quality issues had been done throughout the implementation process. The assessment phase distinguishes new data operations (adding new data to the system) and legacy data quality issues. In assessing on-going data operations, the focus lies in analyzing the processes that lead to the addition of new data and how these can be shaped, monitored and improved to ensure that no poor quality data is added to the system. This is important as it is otherwise useless to do large-scale data cleaning operations on legacy data. Once the flow of new data is under control, we looked at the legacy data and formulated a plan to detect anomalies, prioritize the order of operations and define work packages.

2.1. Assessing data quality

As mentioned earlier in this paper, in the past, most cleaning operations took place on an ad-hoc basis and for a specific purpose. For instance, each year, the university is required to provide information related to publications to different stakeholders. In such cases, the data quality requirements are externally imposed upon the data supplier. However, since such exports only require the university to provide specific types of content, usually particular categories of research output, data cleaning for this specific purpose only affects a relatively small percentage of content in the CRIS. Moreover, since the output format is also defined beforehand, such cleaning tends to be highly targeted and to ignore properties which are not included in the export format. Additionally, the process of cleaning is relatively expensive, as all relevant records need to be examined, one at a time and updated where necessary. This is not only a time-consuming process, it also requires a great deal of domain knowledge from the persons tasked with the improvement operation.

We decided to initially broaden the scope of the data quality effort and to establish some dimensions to help us prioritize the effort. We ended up with four quadrants where one axis expressed the degree of complexity which was involved in cleaning a particular field or record and a second axis that signified the degree of usefulness. While both complexity and usefulness are not simple concepts, the goal of classifying specific cleaning tasks is not to provide a perfect analysis, but a general overview of how these tasks should be approached. In other words, tasks that are highly complex (and thus time-consuming and expensive) and have little use, either immediate or in terms of reusability should have a lower priority than tasks which are simpler and result in cleaner data that can be used or reused. Of course, in some cases, compliance issues may overrule these two dimensions: for example, Pure has an Event metadata object that contains a field “Degree of recognition”. This field is used to indicate whether an event (mostly conferences and workshop) should be considered of regional, national or international importance. By its very nature, Event metadata is highly time-bound and cannot be reused to a great extent once the event has passed. Moreover, finding out more detailed information about events is difficult for whomever is charged with improving the data, as this information is only immediately available to people who were directly involved in the event. To make matters worse, there are few reliable external databases for events, which makes the success of any improvement effort dependent on external factors, such as the availability of the data on the event’s website. While all this factors would place this specific field (and Events in general) squarely in the most negative quadrant, the degree of recognition is used in an internal funding model to distinguish conference contributions at national conferences from conference contributions at international conferences. In this case, an internal compliance issue overrides the aforementioned dimensions and needs to be included in the relevant Events. Obviously, since the motivation for this effort is a specific type of export, we only need to concern ourselves with the Events that are

related to publications that fall within the scope of the export. In other words, historical data, which may be improved later on for completeness' sake, gets a lower priority than the data which will be included in the allocation model. There are other such factors which override the dimensions of usefulness and complexity, especially where external compliance is involved.

The next step we took was to examine the legacy data and analyze specific fields that were somehow suspect. In most cases, this involved identifying missing values, or fields which contained either duplicate or misplaced values due to the migration or poor input in the legacy system. Additionally, on the record level, we also looked at duplicates, as Pure has decent tools to facilitate such an undertaking. Since the system was live at the time, we were also faced with the need to do reactive data quality assessment and improvement as researchers or administrators either signaled shortcomings or asked questions about their data that indirectly pointed us in the direction of data quality issues. Focusing the analysis on the field level, rather than the record level allowed for greater flexibility in allocating resources to perform the actual cleaning. Additionally, it made it far easier to plan the time-investment allowed, as adding data to a single field is generally much more predictable than cleaning a full record, especially for publications other than peer-reviewed journal articles, such as books or chapters in books. Most of the analyses were performed by querying the tables directly rather than using the reporting tools included with Pure. The reason for this is that the reporting module at the time offered limited support querying for missing data points. There were other limitations which made it hard to query relations between different types of data within the application. Detection of duplicates, on the other hand, was largely done via the tools provided by Pure, which not only allows for quick detection, but also offers tools to merge or accept duplicates. This was particularly useful for journals, external organizations and publishers.

These assessment methods do have limitations, in particular that they are most useful for detecting formal data quality issues, i.e. missing or flagrantly invalid values in fields. They are much less suited for tracking qualitative data problems. A few examples to illustrate this limitation: missing page numbers are easily detected but wrong page numbers are nearly impossible to detect, provided they are plausible. Author affiliations are impossible to evaluate, since only the author (and even that is doubtful in many cases) knows what affiliation an external person may have in the context of a particular publication. There are numerous examples where assessors and cleaners may simply lack the information to detect the issues, let alone improve them. In order to try to address this, we tried to raise awareness of data quality issues during our demo sessions.

Slightly overlapping the actual improvement phase is determining the complexity of a particular cleaning operation. To quantify this, we tried as much as possible to simulate how actual cleaning is done: rather than cleaning a small, predefined set of fields or records and extrapolating, we decided on doing a particular operation for a significant amount of time (generally a day) to determine how long cleaning the entire set or a subset thereof would take. This type of analysis takes into account variations, both in the dataset and in the day-to-day operations. Other methods tend to vastly underestimate the time that a certain operation will take. Moreover, working with a larger test sample also allows us to make a more realistic estimate of the domain-specific knowledge needed to accomplish the task and helps determine who could be tasked with performing the cleanup.

2.2. Improving data quality

One of the requirements for selecting a new CRIS was the ability to incorporate data quality workflows, i.e. the solution needed to provide a means to validate new content. Pure does this quite well. Nevertheless, by its very nature, a validation workflow means a relatively big investment in manpower. Moreover, and we probably could have done this better, it is of vital importance to define and document exactly what is being validated and what the desired data quality standard is. At the Vrije Universiteit Brussel, three separate workflows exist: one for research output, one for metadata and one for projects. Since the project workflow has so far been confined to checking financial data and is less concerned with overall data quality standards, we will not dwell on it here. The other two workflows are handled by different groups of people and at a different rate: while some urgency exists in validating research output to ensure that the necessary corrections are actually done and validated, this is far less the case for metadata (it is interesting to note that research output which contains non-validated metadata, such as Events or Journals, can itself be validated in Pure). Also, whereas metadata is generally validated or improved 'behind the

scenes', the validation of research output is a much more interactive process, in which researchers are expected to improve their own publications before sending them back for validation. The dialogue between the validator and the researcher is stored along with the research output in the system for future reference. It is clear that this is a time-intensive process that requires both skill at assessing publication records and tact to communicate shortcoming to researchers who often dislike the registration process itself, let alone an additional validation step. To make matters worse, some issues were only discovered after going live: how do we deal with ephemeral publications, such as a meeting abstract distributed at a conference, should we make abstracts compulsory throughout the system, etc. Our current stance is that the CRIS is primarily used to support the researchers which leads to an attitude where we prefer to err on the side of leniency. While this is a noble goals, it does imply a risk in terms of data quality and it also increases the workload of validators. We may try to resolve this issue later by implementing and communicating clearer guidelines about certain minimum requirements from which no deviation is allowed.

At the time Pure went live, we migrated some 90,000 publications from the legacy system. These automatically got the workflow status 'validated' during the migration. In order to allow researchers to update and improve their publication records, the possibility for revalidation was offered. This allowed researchers to make changes to legacy publications, after which they are proofed by the validators. The alternative was that researchers would have to create tickets or send mails for each change they wanted to make to their existing research output. Since we wanted to encourage researchers to improve their own data as much as possible, we wanted to make the process as painless as possible. To further encourage researchers to update publication data, the decision was made to not retroactively apply validation rules: i.e. if page numbers are required for new publication, but a researcher adds an event to an old publication, it will not be rejected because the page numbers have not been added.

Throughout the demo sessions, it was clear that some researchers were somewhat disappointed with the data quality after the migration. They had also been told that they were not required to do any cleaning after the migration, which is true, since there is no policy requiring them to do so. However, since many of them want to make use of some of the features of Pure (notably, the CV module and the web portal, which opens up the data to external visitors) data quality issues quickly turn to ugly eye-sores. Since cleaning all the data for any one specific researcher or research organization scores low in our prioritization model, we declined doing this for them. To help them out, we do offer assistance to research organizations who want to do a partial or full cleaning project for their data. Generally, this only involves publication data, and we help by providing the organization with listings of possible publications (they are not always marked as such in the older data), along with an analysis of missing values and with guidelines on how they can best undertake a cleanup.

3. Automating data cleaning

From commodifying data cleaning it is only a small step to full-scale automation. Working with a COTS system such as Pure makes it difficult, if not impossible, to make direct changes to the underlying database. However, since the database can be queried, for instance to locate empty fields or other irregularities, it is possible to create lists that can then be fed into scripts that automate some of the more repetitive tasks described in 2. Pure only supports a very limited amount of bulk operations, which means that most bulk operations, such as reclassifying outputs, changing ownership on a large number of records or filling out a specific field with a particular value needs to be done either manually or by browser automation.

The Selenium suite of tools is primarily intended to facilitate testing of web applications. However, since it provides us with a means to generally automate browser interaction, it is also suited to automate inputting values or change records in any web application, including Pure. Selenium has the so-called WebDriver API, which allows its functions to be called from within a number of programming languages. At the VUB, we have chosen to use Python to work with Selenium as it allows for very short development cycles. Nevertheless, Selenium projects can easily be set up in other programming languages, such as Java, Ruby or C#.

A Selenium project follows the same steps every time: 1) create a list of identifiers and (optionally) input values, 2) identify and locate the elements in the web application, 3) write WebDriver calls and wrap them in a script, 4) test the script and 5) run the script. It should be noted that, while Selenium supports headless browsing through PhantomJS, we run the job in a visible browser window. The first step is mostly the same as for manual data

cleaning operations, with the exception that lists need to include all the necessary information for changing the data. Whereas a list for a manual operation could consists of identifiers where the person performing the operation has to assess what has to happen next, this is generally impossible when doing simple browser automation. Selenium does offer the possibility to read, compare and assert values and as such theoretically allows for such an approach, in practice this turns out to prove quite difficult and error-prone for any but the most straight-forward cases. An exception to this is the case where all records in the list undergo the same operation (for instance, being placed in the same category): here, rather than include the relevant value in the input file, we mostly choose to provide it as a variable in the script.

The Selenium API allows for multiple search techniques to locate elements in the interface. We've had most success with XPath statements, as most elements in the Pure interface have dynamic identifiers. There are some exceptions to this rule, such as certain login elements, in this case the identifiers are used, rather than the XPath statements. XPath has the advantage that it is always possible to locate any element, but has the disadvantage that it often locates these elements in relation to other elements on the page. This means that scripts are not robust: depending on how the XPath statement is conceived, small changes in the interface, can result in errors in the script.

As mentioned, we use Python to wrap our WebDriver calls. This is needed to provide a runtime environment to perform the calls, but also to handle errors and, in cases where this is relevant, to write the output of certain operations to a file for further processing. Python also has the advantage that it is easy to learn, so that other people involved in cleaning operations can realistically build their own automation scripts. Of course, when implementing a similar automated cleaning process in an organization that is strongly tied to another language, then it is natural to prefer using another language.

Especially if the script provides robust error handling, it is important to test it in a safe environment. In Pure, changes can only be undone by rolling back the database, which is highly problematic in terms of data loss. Therefore, we generally tend to avoid deleting values while cleaning automated and mostly use it to either populate a single field on a large number of records with input we provide through a file, to change ownership of a list of records (again, we provide identifiers via a file, rather than relying on the algorithm to determine which records to move) or to change the value of a specific field on a predetermined set of records. We are convinced more complex approaches are technically feasible, but they would also involve much more risk-taking, as undoing mistakes would become more difficult.

Finally, when running the script, we use a dedicated computer to do this and try to do this outside business hours. Since we do not do this headless (i.e. all interactions are actually performed in the browser on the screen), working on the same station, even with a second screen, is very difficult.

While this may seem like a lot of hassle, we would like to share the following example to illustrate the amount of time that be gained by automating: after we went live, we noticed that a certain code on projects was migrated into a custom-built field that turned out to be problematic to use. After some analysis, it was discovered that the code should have been stored as a keyword, which would make it searchable. The issue was that this code was present on some 15000 projects and Elsevier was apprehensive to write a query to move the data from one field to the newly defined keyword group. This is understandable, since our institution is probably the only one with this specific customization. Initial tests showed that one person could realistically move 300 codes per day, as getting to the value involved several clicks and page loads. In all, doing it manually seemed simply too expensive for what we would get out of it. After deciding to investigate automating the process, we spent three days figuring out how to do it, writing the script and testing and had the script run for about 24 hours until completion. For later, similar cleanup actions, the development time for such scripts has been reduced to well under a day, the amount of time it takes to run the script is practically irrelevant, since it is always faster than manual cleaning and since it does not tie up people who can do other tasks instead.

4. Conclusions

Migrating to a more comprehensive and transparent CRIS unearthed massive data quality issues at the Vrije Universiteit Brussel. Even after a year and a half of cleaning, much work remains to be done and some parts of the dataset may never be fully improved. However, our experiences taught us that the cost of any kind of cleaning effort

can be controlled and minimized by careful assessment of the problems, by decentralizing a certain amount of the actual cleaning and by commodifying data quality improvement operations. Systematic analyses of data quality issues also allowed for the automation of some of the tasks involved.

We also learnt a number of important lessons which may serve us well in future migrations: first, by restricting user acceptance testing to a small and unchanging body of people, the chance to evangelize the system before it went in production was missed. More importantly in the context of this paper, it also created false expectations with regard to data quality. Second, the lack of consistent data governance in the organization did not merely cause trouble in deciding on taxonomies and classifications, but continues to make it hard to improve certain areas of the CRIS' data. This is particularly the case for Organizations, and to a lesser extent for Persons. While this situation is not new and an awareness that this is a problem does exist, so far all efforts to incorporate this into a broad university-wide data governance plan have failed. This in turn makes it hard, if not impossible, to definitively improve these important data points.

Finally, data quality improvement is just a small portion of an overall data management policy. Without clear data quality standards, it is difficult to create an awareness of the importance of data quality. Without a mature data governance, the owner of one system will often be burdened with poor quality integrated data from other systems. Moreover, the usefulness of one relatively clean dataset in a system is much diminished if the data cannot be adequately exploited because data in other, related systems is of lesser quality or does not conform to the same dimensions. This then often leads to the duplication of data into the system of choice (i.e. the system with the best overall data quality or with the best tools to analyze, visualize or otherwise exploit the data), leading to data which is no longer in sync and the deterioration of the overall data quality in the organization.

References

1. Batini C, Cappiello C, Francalanci C, Maurino A. Methodologies for data quality assessment and improvement. *ACM Comput. Surv.* 2009;**41, 3**: Article 16, 52 p.