Updating the Register

CHARITY COMMISSION
FOR ENGLAND AND WALES

The Charity Commission of England and Wales [https://www.gov.uk/government/organisations/charity-commission] is an independent, non-ministerial government department accountable to Parliament that has the remit to regulate charities.

The official registrar

As the official registrar, the commission is responsible for maintaining an accurate and up-to-date register of charities. Their role includes deciding whether organisations are charitable, removing those that are not, preventing abuse for personal gain, money laundering and the funding of illegal organisations. Data accuracy is essential.

Reg+

The commission's system of record is called Reg+, it sits at the heart of their operations. Reg+ holds the commission's data on the charities and people it regulates. Reg+ maintains a full historic view of the charity, its trustees, personnel, declared purpose, activities and finances from the date it was first registered.

This data is used by the commission's case workers, people who regulate charities, to inform their decision making and to support investigations. The data is also used in the commission's public search and information service [https://www.gov.uk/find-charity-information]. Data is collected into Reg+ when a charity registers, when it files an annual return and when the commission is notified of changes.

Reg+ was to be updated. The commission had three principal reasons for making changes Reg+...

1. They needed to allow charities to maintain information online, improving the quality of the data, as it will be more up to date. To permit this, data had to be switched from periodic collection, as part of an 'Annual Return' submission by the charity, to 'maintained data' that charities are obliged to keep up to date via an online portal.
2. The information model needed to evolve to support better regulation. Changes included:
   - A charity's Trustees are important, they govern the charity and should ensure it 'plays by the rules'. Trustees were to be recorded as individuals, with their own 'identity' and contact details, rather than as names associated with that role for a charity.
   - Information held on individuals who worked for the charities needed to become private, with new public 'corporate' contact information being introduced and published for use by the public.
3. A cleanse was required. The data went back a long way. It had come from various incarnations of up-stream systems, each with differing views and levels of data validation. Consequently, data quality in the individual records was variable. Invalid and questionable data needed to be dropped, forcing charities to renter, validated, data when they used the new online portal.

These demands meant that the Reg+ database schema had to change and the data within the existing schema had to be both transformed to the new schema and cleansed and standardised without losing or harming any of the important knowledge that allows the commission to undertake successful regulation.

Reg+ changes and data transformation

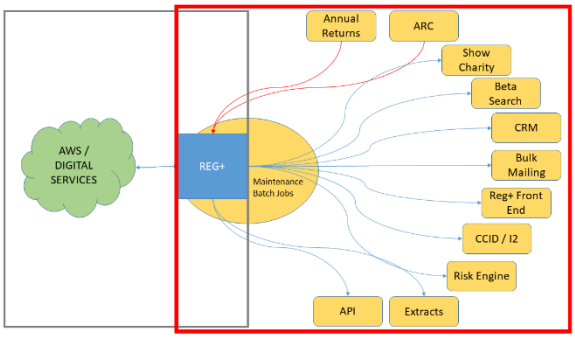
Implementing these business changes meant altering the way Reg+ held data and transforming the data within it.

Reg+ in context

Sitting at the heart of the operational IT estate, Reg+ is key to the day to day operation of the commission

Reg+ is the system of record for the commission. It sits at the heart of the regulatory operations. It is their master system of record and feeds many other systems, both internal and external, that support and regulate the charity world.

The data it holds must be as complete and accurate as possible. The history of that data, who changed what and when, must be reliably retained for audit purposes.



The core changes

At a high-level, the work to be done on Reg+ consisted of…

Schema Changes	The data schema had to be extended to support concepts in the digital portal and to be able to support new maintained data to replace that previously collected annually.
Schema Population	The initial content of new schema elements had to be populated with data derived from the old schema. This process had to eliminate invalid and suspect data, leaving this to be entered by charities when they first used the portal, and it needed to 'normalise' retained data values in fields into standard forms.
Deduplication	At the end of the cycle of deploying the new system and getting charities to enter missing data the commission wanted the issue of duplicate data, holding multiple independent, possibly differing, records for an individual or entity, to have been eliminated. This required a process that identified duplicates and either resolved them to a single, logically equivalent, record or removed them to force re-entry.

The additional stuff

This work on Reg+ then meant some supporting changes also had to be made to the…

Audit History	Recording the history of data is very important to Reg+. Knowing what value a record held yesterday, and at times before that, is as important as its current value. Much of main schema is versioned to provide this. However, over and above this, each table has a parallel audit table that has a record for each event that occurred on each row in its master table. The audit trail needed to be retained and had to continue working.
Updates to Feeds	As the 'system of record', Reg+ fed information to around ten downstream systems. These feeds needed to be updated so that, going forward, they would take their data from the new areas of the data schema. These changes needed to be transparent to the downstream systems.

Managing the risk

The Commissions Head of Information Services and Programme Test Lead recognised that there was a significant risk that the Reg+ changes could be problematic, and they engaged SQC to perform technical data migration testing.

Why the concern?

Data migration and transformation is both inherently error prone and intrinsically hard to test, see "**What is hard, what can go wrong?**" at the end of this case study for some technical details. The issues include:

- Transformation rules can be complex to conceive and express and hard to implement. Decisions may be applicable in general but fail to address complex or edge cases.
- Data, created over time, can be 'dirty'. It does not match expectations that are derived from how the source system works today, often we find data that was created a long time ago when things were different.
- Inspecting and validating what has been produced is challenging. Traditional testing, using the systems that will process the data, provides views that are one or two steps removed from the detail.
- Validation must be comprehensive, every record needs to be checked, again this is not practical using traditional testing methods.

Why SQC?

SQC had already undertaken technical testing for the commission, delivering load and performance testing on a number of digital projects. The commission had got to know and understand SQC and were aware of their specialist technical and assurance capabilities. They recognised there were key factors that made SQC the right organisation to undertake this work. These included:

- Autonomy, shaping and driving the delivery of the work, rather than relying on, very busy, commission teams to make things happen.
- Rigorous analysis, bringing an approach that incorporates a detailed risk analysis as the basis for scoping and prioritising the testing work to be done.
- Rigour in implementation, an approach that used comprehensive 'deep inspection' and reconciliation of the source and output data sets as the principle validation method, rather than sample-based testing using normal systems.
- Technical competence, the ability to work with the data sets and develop the tooling needed to perform the testing without requiring support from the various development teams involved.
- Constructive critical thinking, a willingness to challenge the assumptions of the commission and the development team regarding what were the 'right things to do' when transforming the data.
- Pragmatism, a pragmatic, iterative, approach to the development of understanding of the data space and implementation of the tests. Recognising that there would be a learning process for both teams.
- Flexibility, recognition that this work could not be specified and planned in a 'safe' deterministic way and that the way forward would need to be assessed as knowledge developed.
- Responsiveness, the ability to respond quickly, to start when the commission needed the work to start, to adjust to the pace of the commission's work and to respond quickly to changes.

How to test a data migration

The testing of data migrations and transformations is a specialist discipline. An approach that uses generic testing practices, those normally used for functional testing, has a limited chance of success. Something extra is required.

Forensic testing

Ensuring the integrity of the results of a migration demands fine grained analysis of the data content. Comprehensive reconciliation back to the source data is essential, as is a detailed quantitative analysis of the impacts of data cleaning on the content.

Feeds to downstream systems needed to be compared, record by record, to demonstrate equivalence. The risk that feeds pick up data from 'old locations' rather than new ones means that changes have to be applied and witnessed in both full and delta feeds.

The testing needs to be detailed rigorous and comprehensive, focussed on the contents of the database and of the feeds, rather than on the viewing data through the lenses of systems that operated on the data. The test team must be able to conceive the mesh of data centric tests that will detect anomalies hidden deep within a large and complex set. They also need the development and database skills required to implement these tests effectively.

Why an iterative, adaptive test delivery approach is essential

The definition and implementation of the tests must be guided by a detailed understanding of the data model, of the real-world characteristics it represents and of the business requirements for the migration. This is a complex arena of knowledge, not one easily obtained from typical specification and design documentation. The vast gap that exists between an abstract description of a data migration and the reality of handling a complex, and dirty, data set means there is a clear advantage in an iterative adaptive test approach to evolving both the test definitions and test implementations through "learning by doing".

It makes no sense to write numerous detailed tests, a monolithic test specification, without first having explored the data and observed the early versions of the migration process in operation. The nuances of transformation are difficult to grasp without a firm understanding of the data and some understandings only emerge when the outputs can be analysed. The devil is in the detail. Writing tests upfront will result in many flawed tests and many gaps in the test set.

The best answer is to 'get your hands dirty' as soon as possible. To explore the problem space, find things early and to let the formal tests emerge from this process. This is adaptive, exploratory led, testing, an approach that finds high value issues early, builds a lean tailored test set and avoids unnecessary rework on tests.

Mistakes and anti-patterns this avoids

This approach avoids common mistakes made when organisations test data migrations.

- It does not rely on the operation of consuming systems to check the contents of the data set and of feeds drawn from it. Many checks are done directly on the data.
- It does not attempt to write a monolithic set of tests upfront, based on specifications and data models. Instead, after a preliminary risk assessment and test outlining exercise, it takes each test objective in turn and works out what really must be checked in conjunction with examining and running tests on the data.
- It distrusts documentation, both the documentation of the system and the specification for the migration. It questions everything and the thing it trusts most is the data.
- It does not assume that you can get the testing "right first time", not "complete first time". Rather it envisages there will be a number of cycles of testing with the testing improving on each cycle.

Applying this to the testing of Reg+

In the case of Reg+ this approach to testing proved to be a good fit and produced very valuable results.

The test activities

When tailored the test approach the sequence of principal activities that emerged was:

- A timeboxed risk assessment and strawman test requirements creation exercise. We outlined potential threats, failures, and tests. Outlined rather than specified in full.
- Exploration of the data, developing an understanding of the data and the migration approach using a copy of the production database and a migrated database created by the first versions of the migration processes.
- Ad-hoc fulfilment of the data content test requirements, creating query sets that implemented each type of test requirement. We learnt and evolved the test techniques, we revised the test list.
- Reassessment of test requirement adequacy,
- Industrialisation and roll-out of the tests across the whole dataspace
- Generating and reconciling the contents of data feeds from the pre-migrated database with the same feed generated on the migrated database.
- Confirming that changes applied to the migrated database were reflected in the feeds contents of the feeds coming from that database.

The technique worked well and suited the culture and practices of the development organisation. It caused us to ask for answers to specific, concrete, questions rather than seeking an all-encompassing explanation at the start. It worked well with the incremental, and iterative, way the migration was shaped and implemented.

Different ways we implemented tests

A data content test set of around one hundred and fifty specific reconciliation checks emerged during the process. The checks ranged from trivial row count comparisons to highly complex queries running across both databases. We created around 2000 lines of reconciliation SQL implementing these tests.

For the audit tables we implemented an SQL generator in C# that took the meta data on the table and its associated audit table and generated queries that ran across the four tables to validate that the history reflected the migration actions. This generator created around 6000 lines of SQL to provide a comprehensive check of audit behaviour.

Downstream system interfaces came in a variety of forms, the main ones being database views and SQL scripts that created extract tables or files. Hand coded SQL harnesses where used to compare the contents of database views. PowerShell harnesses executed SQL extract scripts and then ran comparison queries across the results generated for premigration and migrated versions of the database.

A very fluid situation

Testing did not simply "test an implementation", rather it helped to shape the entire migration. Questions raised when assessing an area and anomalies revealed when an area was executed and analysed crystallised factors that, until then, had not been explicitly recognised. This drove wholesale change to certain aspects of the migration approach.

The migration itself became a fluid concept, responding and evolving in the face of the findings of the migration testing. Consequently, the migration testing itself had to be very fluid. Responding to its own findings and to the changes in the migration concept and approach it drove. Only the adaptive approach that had been adopted could cope with this fluidity, a rigid upfront test specification approach would have been broken.

What value did the testing provide?

When we started there may have been perception, amongst some of the development team, that this was an unnecessary exercise. One that would add little value given the testing that the development team would be doing. If this perception did exist, then it proved to be very wide of the mark.

Was it worth the effort?

The testing uncovered a significant number of issues at all levels. Issues that ran from simple implementation errors through to debates on whether business requirements were sound and around what the policy should be on data cleansing.

The testing has a material, beneficial, effect on the process. Without it there would have been major problems. It was clear that, if this testing had not been done, then these issues would either have disrupted integration testing and user acceptance testing or, worse, slipped through the net into production causing operational problems and recovery costs for the commission.

What the testing revealed?

A broad range of issues, of many different types, were discovered by the testing process. Some were simple development mistakes, whilst, at the other extreme, some identified gaps in, or challenged, the decisions made by the commission on how to handle and associate data. To the test team the number and variety of issues did not come as a surprise, though many individual issues could not be predicted as they were very much related to the detailed nature of the data. For the delivery team, however, the volume did come as somewhat of a surprise.

The nature of the issues

As described above, there were a lot of issues found and they took many forms. Too many and too varied to list all the different types. That said, a flavour of what went on and what was discovered can be gained from the examples given below. This list attempts to show the different categories of things that had to be tackled.

Implementation errors and omissions	There were examples of well understood concepts, ones to be applied during the migration, not working properly. Sometimes due to the "noisy" nature of real-world data and sometime due to simple human errors that had slipped through development team testing. Examples were: • Not recognising there were conflicting records and quarantining the conflicts. • Dropping records, as if there were a conflict, when there were no conflicts. • Migration routines that exploded memory utilisation and exceeded database resource limits. • Aspects of the agreed migration approach that had simply not been implemented • Columns and triggers missing from data change audit mechanisms. • Interfaces that were meant to output identical data before and after the migration that produced materially different outputs. • Interfaces that needed to pick up and propagate changes made to the data as part of migration that failed to detect and transfer the changes. • Duplicate records in interface output caused by errors in modification to the interface implementation made to accommodate data schema changes.

Reg+ Data Migration Testing

Over simplified implementation of rules	Real-world variants and edge cases can make even the most "basic" business rules complex to implement. Candidate implementations can "gloss over" this complexity, something that does not get recognised until the implementations face rigorous testing. Comparison of telephone numbers was one case encountered that illustrates the point. A literal comparison may not mean the true business needs. For example, there are multiple sequences of digits and symbols that can, legitimately, represent the same phone number, even more when spaces come into play. Matching that truly mimics real world equivalence required a more sophisticated approach than was originally implemented. It took several iterations to converge on an acceptable solution.
Intentional bonus changes	Testing spotted discrepancies between what the migrated system would have sent on external interfaces and what the new implementation sent. Analysis then traced these back to changes in the logic applied to select and filter downstream interface data. These changes turned out to be "improvements" that the development team thought would have no effect on the externally visible behaviour of the system. However, as always in these situations, the complexity and diversity of the data tripped them up and small, but significant, anomalies were introduced.
Unrecognised side effects of requested changes	High level agreements, specifying how to move information from the organisational level to individuals, had unforeseen consequences. These only surfaced when the migration testing highlighted the impact in the data set and output data produced by the migration. We found organisations, previously contactable, that now had no channels through which they could be contacted.
Over simplified rules	Identification of "over simplified" requirements, requests for data changes that were, on the surface, easy to define and ask for, that became far more difficult once the number of problems the "simple transformation" created was surfaced during testing. Just as implementations can ignore complexity, the requirements provided to the team van also fail to address the full range of situations that will be found in the real world. The sorts of issues that had to be explored included: • What to do with addresses where the postcode was once valid but has now been retired by the post office. This was something not considered by rules relating to the postcodes being "valid". • How to classify accounts held with financial institutions that had changed from mutual building societies to a commercial banks or where institutions had merged.
Ownership of enabling activities	The testing had to drive the execution of enabling activities. It turned these from vague "things we know we will have to do at some point" into concrete tasks that people had to work out how to do and then get people to do. These items often fall through the cracks, only getting done at the last minute. This creates problems as it is then discovered, late, that these tasks are poorly conceived or defined, are difficult to do and are found to depend on other tasks and changes that nobody had recognised until that point. The types of things in this category include: • Updates to reference data, such as list of countries, that need to be made to achieve data quality objectives. • Cleansing to anomalies in the production data that are "tolerated" in the existing solution but that cause issues for the new one.

What is hard, what can go wrong?

Data migrations and transformations are complex and, if not done very carefully, error prone.

Hidden Complexity

Often, on the surface, summarised as a list of changes, migration and transformation work does not sound too complex, challenging and risky. People are misled because most of the issues that will be encountered stem from the contents of the data, frequently from the contents of small subsets within it, or in the depth of the algorithms that will be needed. This complexity is masked by the high-level descriptions of what is to be done

An introductory illustration of a 'simple' request that hides complexity	*"Compare the telephone numbers for a person and determine if they have more than one actual number", was one of the requirements. It sounds simple, until you look at all the different forms people have been able to enter telephone numbers in over the years. Until you realise sometimes there are two numbers in one field.* *The challenge is to achieve what the business wants, no missed cases but no flagging multiple numbers when all the strings represent the same number in different forms. It is easy to do a direct compare, but that is wrong and the logic to 'do it right' can be complex and, hence, error prone* *. When the output of testing with the real data set were analysed we found cases were multiple numbers were not flagged and cases where identical numbers were flagged as multiple.*

Dirty data

In the real world, data is updated by software with bugs in it, batch processes are run twice and data patches go wrong. Not only that, the rules defining what is correct tighten and change over time. The result is the data in the database often does not align with what it would be if I were created today. If not spotted, this can cause unforeseen, often hidden, consequences when a data transformation process is executed. If spotted, it adds complexity to implementation and testing. Some types of dirty data issue are descried below...

Duplicate records and rows	Where, nominally, there should have been one record or one row with a given set of column values, we find two or more. Such states should not have been possible, the way the software operated should have prevented it, but it did not.
Broken references	Fields that should contain values that match ones in another table have invalid values
Malformed values	Simple values are often malformed. Email addresses without a domain, or with spaces in the body. Leading and trailing whitespace.
Obsolete data	Data collected over time can be obsolete, but when people specify transformation rules they consider contemporary data, not what it might have been ten years ago. Examples include postcodes that are no longer in use and account information from building societies that have now turned into banks.
Null records and mandatory fields	Databases can be littered with rows that have been created, but then did not go on to have meaningful values set in columns.

Transformation demands

Data is not simply copied, 'as-is', from place to place. Analysis, filtering, grouping, adjustments, merging and partitioning tasks are often required. Again, though simple to describe, these tasks are made complex by the nature of the data they must process. Some examples are given below...

Creating structured data from free format fields	There is always a challenge where free format text fields have been used for non-narrative purposes and the system is changed to use restricted, specified, values for the same purpose or to use a structured storage for a composite record. There is likely to have been little validation, aside from length, and the field will contain lots of variation of valid data and quite a bit of nonsensical data. Automatically mapping values to the write restricted value or structured information and avoiding throwing away basically 'good' data because it isn't matched is a challenge.
Matching	Most migrations and transformation from one schema to another of from a legacy system to a new system involve matching of entities. People, postal addresses, telephone details, organisations, often must be matched. Correct matching is not trivial. For example, the CRM systems of telecoms or similar consumer facing companies are well known of having multiple, non-identical, contact records that represent the same person out there in the wild.
Merging	Matching is one thing, often when matched things need to be merged. A common objective here is to merge multiple accounts, perhaps from multiple systems, for a matched person into a single account on a new, all encompassing, system, Incorrect matching is a source of one sort of merging issues, but, on top of this merging is inherently complex and error prone.
Repartitioning	Some of the most complex challenges come when data must be repartitioned. For example, taking data from buckets based on anniversary dates and reallocating it into buckets based on calendar period dates.

Data history

Systems that maintain a data history have an additional set of challenges.

Versioned data	The main schema contained versioned and data stamped information. System pages and reports could show who things had changed over time. When the way information is modelled in the system is refactored then creating an equivalent history is complex.
Audit trail	Audit tables provide the full history of the creation-of, updates-to, and deletion-of rows in master tables. Grasping what the correct record should be when a, one-off, transformation is applied and ensuring that this is the case is not trivial.

The other challenge, spotting the errors

On top of all the complex issues involved in getting the transformation right, there is another big issue. Spotting when it is not right. Transformations tend to product complex data sets. Concrete forms, rows in tables, are not always easily viewable in a logical form, business information. Problems can be hidden in the fine detail of the information or as latent issues in the way that is represented in the concrete form. It is hard to see into the data using the systems that normally interact with it. Deep data inspection is required and that is not trivial.

Want to know more?

To find out more about technical testing services provided by SQC please visit the technical assurance section of our website [https://www.sqc.co.uk/perform/assurance/technical-assurance] or email or enquiry mailbox [enquiry@sqc.co.uk] and we will get back to you.

What else does SQC do?

SQC has been involved in ensuring the success of complex and demanding IT and software development projects since the early 1990s. The team specialises in the direction and assurance development and delivery activities. The unusual feature of SQC's offering is its foundation on a deep technical competency. This competency combined with delivery leadership and testing competencies provides for:

- Leadership and direction of programmes that understands the engineering involved, that is not content agnostic.
- Assurance and assessment of programmes that can look into the future and identify the complexity that has not been surfaced or declared.
- High intensity, comprehensive, rigorous functional, technical and acceptance testing that shows the true state of development and fitness for purpose of solutions, identifies the full set of things that have to be done to complete the work and, ultimately, drives quality up.
- Development and operation of test support tools and test automation solutions.
- Assessment of how organisations work. How they develop software, test software and deliver programmes.
- Transformation of development, test and delivery organisations.