Merging and its procedures in QSR software

Clare Tagg
Tagg Oram Partnership

Abstract

QSR have provided Merge tools for combining NUD*IST or NVivo projects. These tools appear to offer team research projects the option to develop work separately and combine it for cross-analysis. Although the Merge tools are easy to use, effective use imposes some limitations on the use of QSR software. This paper discusses four strategies for using Merge effectively in team research.

Introduction

Qualitative software is often seen as an important tool in managing large team-based qualitative research projects. QSR provide a Merge tool specifically for projects involving more than one person. This paper examines the procedures involved in using the Merge tool in team research. It is based on the author's experience of providing consultancy support to many varied team research projects using QSR software.

Unfortunately, none of the QSR range of software is multi-user. The software expects all the data for a piece of research to be kept in a single 'project'; actually a collection of interrelated files in a folder. This project can only be accessed by one person at a time even if the project is on a local area network. Moreover, many team projects involve researchers working at different locations either because they are out in the field collecting data or because a number of institutions are collaborating. To solve this problem, QSR have produced Merge software for both the NUD*IST range (N4/N5/N6) and NVivo. This software basically takes two separate projects and produces a new, combined project. More than two projects can be merged by repeatedly running the Merge program. This allows individuals to work on their own project and use Merge to combine their work.

QSR Merge Software

The Merge software has no innate knowledge of the projects; it just combines them by following a set of rules. In general, objects (eg documents or nodes) in the project that are deemed by the software to be the same are combined and objects which are different are just copied to the resulting project. So if there is a node in each project with the same name and address, this node will be in the combined project with all the coding from both projects. The manuals for the software (only on the CD-ROM for NUD*IST merge) provide extremely thorough descriptions of the merge rules and advice on how to use the products. The two projects must be of the same type before merging; it is not possible to merge a NUD*IST project with an NVivo project without converting the NUD*IST project to NVivo first. However, N4, N5 and N6 projects may all be merged with each other without conversion. The rules used for merging, and the amount of information provided by the software on the process, vary between the two products.

NUD*IST Merge is provided free with the software (it is actually on the CD-ROM but requires separate installation). It also allows N4 projects to be translated between Mac and PC. The program provides no options or interaction during use. You simply specify the two projects to be merged and a location for the result and click Merge. Documents are merged if they have the same name, same number of text units and their sections fall in the same place. The text in the text units does not have

to be the same; the text for the document in the combined project is taken from the first project specified. Coding for matched documents is combined additively from both projects. Unmatched documents from both projects are just copied into the combined project complete with coding; names of documents from the second project are adjusted if they clash with documents in the first project. Nodes are merged if they have the same address regardless of their name. The name and description of matched nodes is taken from the first project and coding is taken from both projects. Unmatched nodes from both projects are copied complete with coding to the combined project. While matching on node address makes allowance for typos in node names and other discrepancies it can, however, mean that completely different nodes risk getting merged. This is particularly dangerous for free and search nodes.

Merge for NVivo is a much more sophisticated program released as a separate package in September 2001. Merging NVivo projects is technically more complex because of the many more different types of data in a project and the software recognises this by providing an alignment step. During this step the two projects are compared and an aligned project is produced. Merge for NVivo signals that the two projects being merged do not have the same status and calls them the Master and Auxiliary (these behave in a similar way to the first and second projects in NUD*IST Merge). In alignment, it is the Auxiliary project that is converted into an aligned form. As with NUD*IST Merge the project is not altered but a new project is produced. Merge for NVivo may be stopped at this point and the aligned project saved for investigation, or it can be merged with the Master project to form a new project. The dialogue also allows this new project to be merged with further projects making it very straightforward to merge many projects.

During the alignment of NVivo projects the user can decide what action should be taken regarding objects in the two projects that nearly match (skip the object, rename/relocate the object or abandon the alignment). A summary and detailed log is produced of the alignment process. Merge for NVivo considers all aspects of an NVivo project (team members, documents including memos, nodes including coding, attributes, sets, doclinks, nodelinks, internal databites, external databites and models). The rules and level of merging vary but the combined project always contains all items from the Master project and some from the Auxiliary project. As with NUD*IST Merge the program has rules for each type of object which depend on whether objects are deemed to match, but the matching rules are different and more complex. Key differences are that documents only match if the number of characters, paragraphs, in-text links and internal databites are the same. Nodes only match if they have the same full name and address (tree nodes only). Search nodes are treated as a special case; descriptions also have to be the same for nodes to match. These rules mean that it is far less likely that nodes representing completely different ideas are unintentionally merged but they do have the consequence that a misplaced space in a node name, for example, will prevent two similar nodes being merged.

Both Merge programs are easy to run but difficult to use because it is hard to ensure that the right objects match. The ease with which documents may be changed, particularly in NVivo, means that it is very possible for a document in two projects that started off the same to become different. Similarly two researchers may start off with the same node tree but will want to add and move nodes as they code and the trees will become different. In NUD*IST Merge there is the danger that the two researchers will have nodes with the same address but completely different meanings. In Merge for NVivo, the problem is that even if the two researchers discuss their new nodes and agree on addresses and names for them it will be very easy for the names to be slightly different and so not match. The very facilities that give QSR software its power in qualitative research make it difficult to merge projects.

Both versions of Merge are easy to use and are essential tools for those using QSR software in team research. Merge for NVivo is more sophisticated but in the main this is necessary because of its technical complexity. Although the alignment tool provides some useful information it does not tell you why things do not match when you think they should. For example, if two documents have the same name but do not match and you think they should, you probably want to know exactly where

they differ so that you can manually make them the same. Unfortunately the alignment log only tells you that the character count is different.

Strategies for using QSR Merge

During work with teams using Merge software a number of strategies have evolved that can be adopted to alleviate these problems. The choice depends on the type of research being undertaken and the nature of the teamwork. In practice, projects often use a combination of strategies or variations on a general approach, but for simplicity the following four strategies are described separately. The first two strategies apply where a project is being undertaken by a team because it is either too large or has to be done too quickly for one person to complete it. Typical of these kinds of projects are evaluations across multiple sites or the analysis of responses to time-critical consultation exercises. The other two strategies apply where there are essentially several pieces of research being undertaken but some cross analysis is also required. Case study based projects often fall into this category where different institutions are undertaking self-contained studies but the research design calls for some cross-case analysis towards the end of the research. There are also projects where the same data is being analysed from a variety of perspectives by different researchers.

The first strategy, one master project with separate data collectors, is appropriate for a large project involving many field workers. This approach has been used on a substantial evaluation of legal reform that involved interviews with stakeholders in different parts of the country and separate field worker responsible for each area. The documents were collected and imported by each field worker into their own auxiliary project. These were periodically merged into the master project where the major analysis was undertaken.

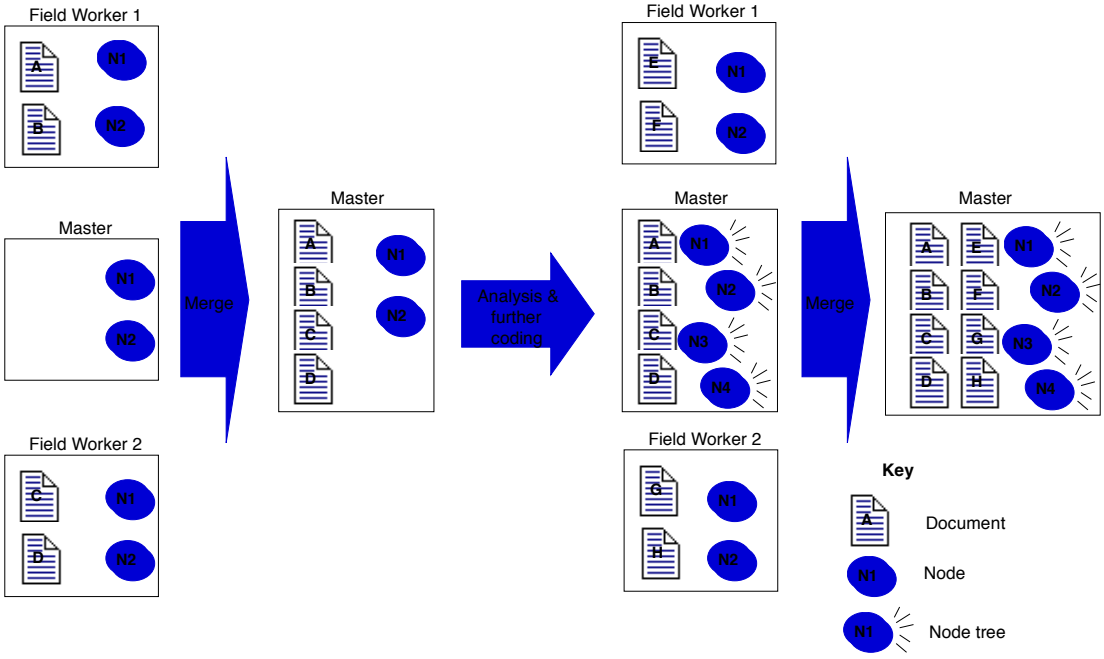

Figure 1: Master project with separate data collectors

This strategy means that when merging occurs there are no matching of documents. In this type of project it can be very helpful if field workers also input attribute values (base data coding in NUD*IST) and possibly do some initial coding. For this to merge properly the node tree and attributes must be defined the same way in each auxiliary project. N6 provides a task constructor for

this, but the easiest way in all other versions is to create a template project, which is copied to form the basis of each auxiliary project and the master project. If appropriate, field workers can be given their own part of the tree where they can develop nodes during their coding. These nodes can form the basis for discussion during the periodic merge. After merging it is important that either each field worker is given an empty auxiliary project to work on or at the next merge the auxiliary projects are merged into an empty master project. If this precaution is not taken, it is likely that the field workers will alter either coding or documents that have already been merged and these changes will conflict when merging is repeated.

One of the problems with the above strategy is that each researcher cannot use the power of the software to develop nodes and a node structure as they code, it has to be predefined. This is particularly problematic when multiple researchers are being used because time is short. In this situation there is generally insufficient time to develop a mature coding structure by negotiation. An alternative strategy that can work well is one master project with separate coders each coding parts of all documents.

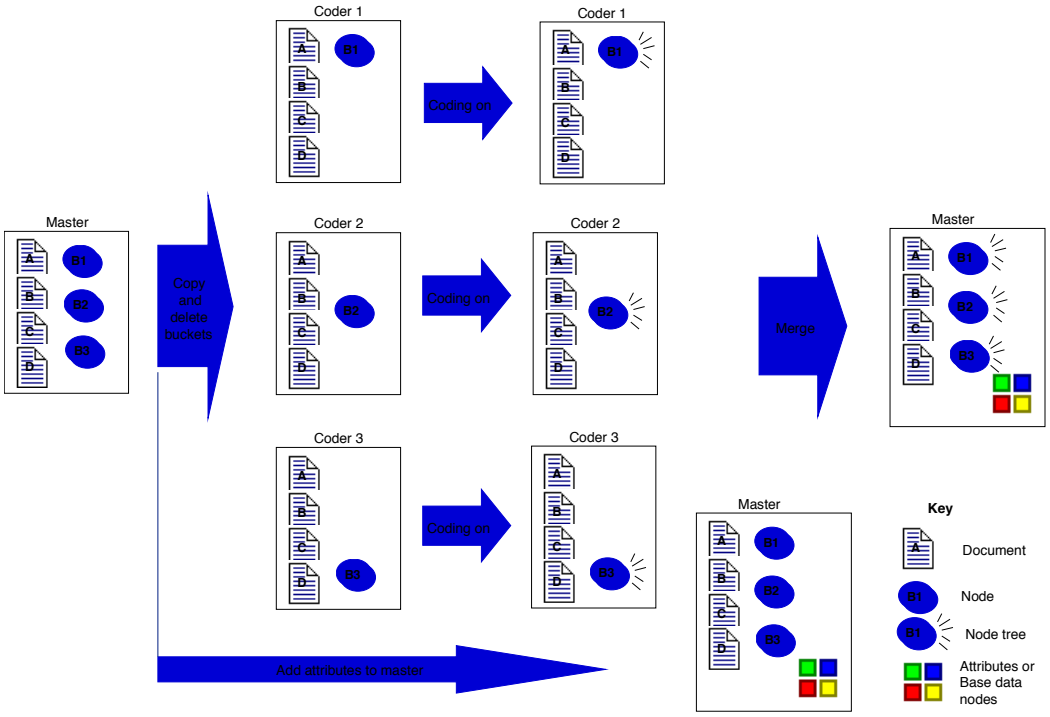

Figure 2: Master project with separate coders

The easiest way to set this up is for all documents to be imported and coded to 'bucket' nodes. A bucket is a broad area of interest in the research; typically there will be between 5 and 15 bucket nodes. This initial bucket coding can either be undertaken by one person or can be done as part of document preparation by inserting codes into the text. Text searching or section coding can be used to automatically code to the bucket nodes. An auxiliary project is created for each researcher containing all documents but only one of the bucket nodes (by copying the project and deleting the unwanted items). This researcher can then 'code on' from this node and develop a series of nodes based on a refinement of the bucket node.

The coded auxiliary projects are merged into a master that contains all the documents and attributes or base data coding. Full investigations can be undertaken on the merged project but investigations within the scope of a bucket can be undertaken on the auxiliary projects. When coding, researchers

have complete freedom to develop the coding hierarchy within the bucket node but must not amend or annotate documents otherwise the documents will not match on merging. It is simplest to organise this approach if document collection occurs before coding starts but it is possible to combine this with incremental document import. This strategy has the advantage that one person sees the full range of documents but can only be used where the research question is such that a small number of largely discrete bucket codes can be defined up front.

A project that has used this multiple coder approach effectively involved the evaluation by interview of business processes within an organisation during a large high visibility project. The bucket nodes related to stages of the project and relationships with other organisations and reflected sections in each interview so that bucket coding was automatic. Four people undertook the coding of a group of related buckets but during the coding on, identified additional text relevant to one of the other buckets where it could be coded by the coder for that bucket. This enabled the coding to be completed in a very short time but did involve a complex series of merges and creation of auxiliary projects.

The third strategy is much simpler and applies where there are several parallel projects being developed, possibly by different institutions, with a limited amount of cross analysis. Each project develops its own master project covering their case and these master projects are merged for cross analysis. If merging occurs subsequently then a new combined project is created. This approach was adopted for research involving the investigation of interaction and relationships between different health-care bodies.

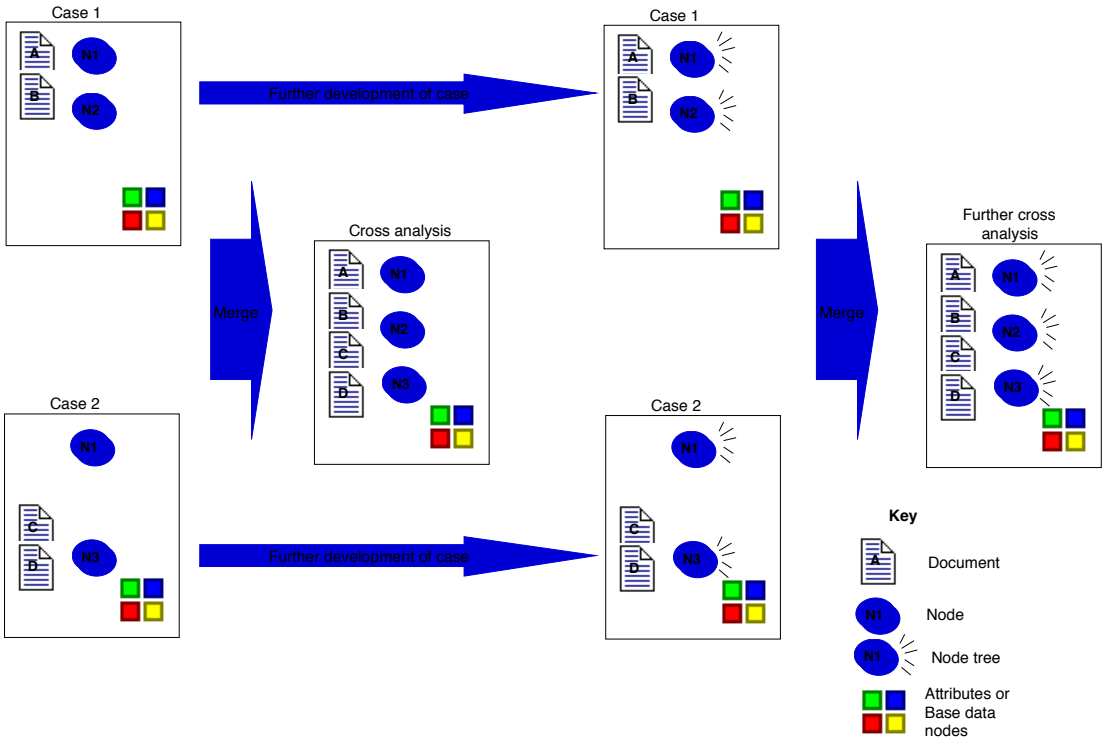

Figure 3: Multiple master projects for separate cases

To facilitate the cross analysis it is important to agree on project structure (eg naming conventions for documents and nodes) so that unlike objects do not get merged accidentally (particularly likely in NUD*IST Merge). It is also helpful to agree on areas where cross analysis is likely to be required. This particularly applies to attribute names and values (or base data coding) but may also apply to some nodes as well. In fact discussion of these issues early in the life of the project can be very

helpful in shaping the research and providing some unity of purpose. Without these kind of initial discussions and agreements it is unlikely that merging the projects will produce many insights.

The final strategy is similar but is useful where the whole dataset is being used for multiple purposes, perhaps by researchers working in different disciplines, but there is a possibility that some cross analysis may be undertaken. A typical project involves the investigation by two researchers from different disciplines of the relationships between two groups of professionals working in government departments. In this case each researcher is maintaining their own project and developing their own coding structure but the documents appear in both projects. Occasional merging of the two projects will allow them to undertake cross analysis.

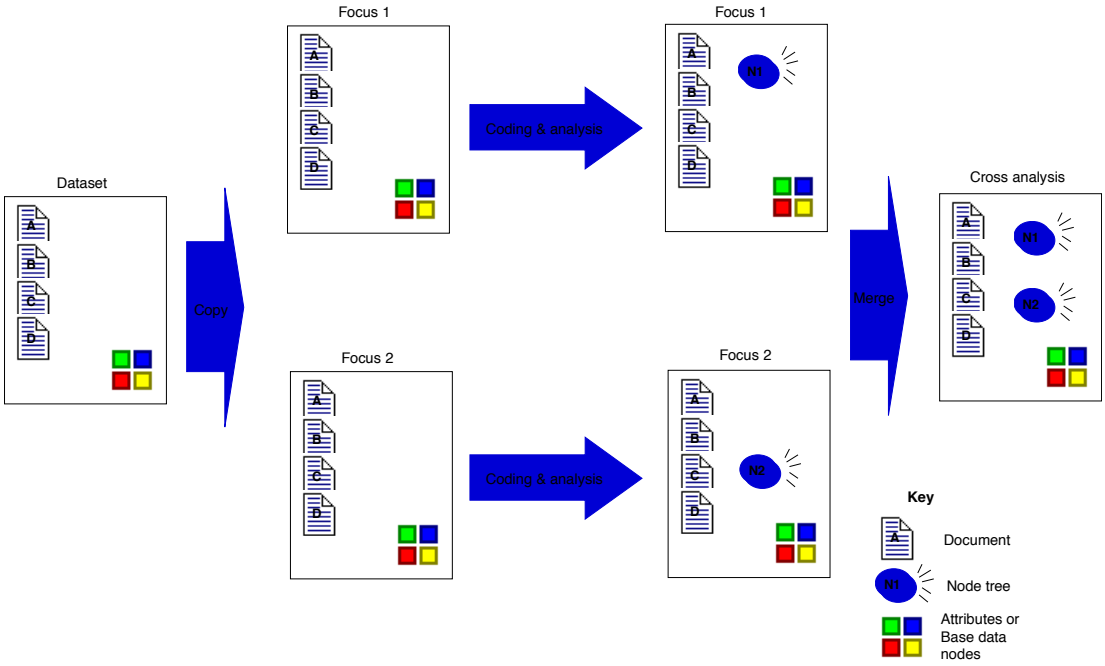

Figure 4: Multiple master projects with separate foci

An agreement on a naming convention for nodes is necessary to ensure that if projects are merged at a later stage no unwanted matching occurs. It is also important that neither researcher alters the documents. This strategy can be achieved by starting with a project containing all the documents and any automatic coding and attribute values, copying it for each researcher. However, if the data is being collected in parallel with analysis, an auxiliary project can be created just containing the new documents with automatic coding. This is merged into each researcher's master project. (In NUD*IST it may be simpler to just run a set of command files against each master project.)

Merge software may also be used to combine two separate pieces of research and produce some comparative results. This is likely to be difficult because of differences in the coding trees. Probably the easiest way to approach this is to keep the coding trees separate and then use search tools to combine selected nodes. This can easily be achieved by ensuring node addresses in NUD*IST or names in NVivo are unique. Merge is also useful for recovering data from backup copies of projects. If a node or document is accidentally deleted from a project, it can be restored from a backup by removing from a copy of the backup all but the missing document or node and running Merge.

Conclusion

Using Merge effectively places limitations on the ways in which QSR software may be used. These limitations are necessary if objects are to be matched during merging. Objects need to match if information from two projects is to be compared using the full capabilities of the software. For example, to compare how people at different case study sites talk about an issue, the same node structure needs to be used for that issue for both case studies. To investigate the connections between coding by different researchers looking at alternative perspectives on a research problem, the same documents, coded by each researcher in their own project, need to match. The four strategies outlined above provide examples of how these limitations may be minimised. The choice of strategy will often depend on the nature of the research project and team structure.

However, as the diagrams show, managing the procedures for creating and merging projects requires good organisation. The most successful projects appoint a single person to be the guardian of the merging process. This person needs to have good computer skills to be able to move projects between machines, keep track of the many different versions of projects and, in the case of NVivo, ensure that internal and external project names are kept consistent. They also need to be aware of the objectives of the merge process and to check before merging that researchers have heeded the necessary limitations and after merging that the results are as expected. The alignment tool in Merge for NVivo can help with this checking.

In conclusion, the Merge software is easy to use and does allow teams to work together using QSR software. However, it is not an easy option and effective use requires careful planning and discipline. A multi-user version of the software would be much easier for teams working on a local area network.