



Ordering concepts

In: The NVIVO Qualitative Project Book

By: Patricia Bazeley & Lyn Richards

Pub. Date: 2011

Access Date: May 25, 2020

Publishing Company: SAGE Publications, Ltd

City: London

Print ISBN: 9780761970002

Online ISBN: 9780857020079

DOI: <https://dx.doi.org/10.4135/9780857020079>

Print pages: 113-131

© 2000 SAGE Publications, Ltd All Rights Reserved.

This PDF has been generated from SAGE Research Methods. Please note that the pagination of the online version will vary from the pagination of the print book.

Ordering concepts

In the context of the earlier exploratory Parts, the tools for managing nodes in hierarchical trees are just one way of shaping data. You will rapidly find however that this is a special way. NVivo does not require that you order nodes, but nodes almost always have a logical relation, and sometimes a very strong one. A logical organization of the nodes serves to clarify the extent and focus of the project, and the emerging structure of the ideas being generated or tested through the project. It allows you to find nodes easily and use them reliably. As ideas change, grow or become more refined, a tree structure for holding those ideas can reflect those changes. As with the shaping tools in the earlier Parts, the tools for ordering concepts are designed to be used constantly and in context of coding, memoing, reflecting, browsing, and editing. This Part is designed to help ensure that you get the structure of your concepts and categories into the kind of order that will allow you to work effectively with your data. Many of its tools are, by now, familiar—yet there are new ways of using them to explore, clarify and refine, as you move toward the goal of fresh insight and new understanding.



Thinking trees

The idea of placing nodes in trees was introduced earlier primarily as a way of setting out subcategories or dimensions of a concept, with trees being created and used on a relatively *ad hoc* basis. As these develop, most researchers wish to organize nodes systematically into conceptual trees, or a hierarchical index system, much as a library catalogues its books. In creating trees and managing them, you clarify your ideas about what goes with what, see missing categories, and solidify categories that overlap. Thus they will be informed by, and in turn will inform the analytical and theory building processes of the researcher.

It may be that in your project, much of the work covered in this Part is already done. It is, however, our experience that far too often researchers struggle on, encumbered by a confused or cluttered index system that inhibits rather than enhances their analysis. Much can be seen about a project simply by reviewing its index system. Usually, a “healthy” well-conceptualized and well-organized index system, or tree structure, is a sign of a project that is progressing well toward a conclusion.

All qualitative methods—and all human thinking!—involve abstracting from the particular to the general. The ability to see a particular concept (e.g. “academic family”) as one of a grouping (“family background”) helps the researcher manage concepts and think more abstractly. (Now we can wonder how different family backgrounds pattern experience.) If this concept ordering process is new to you, try adding a further question to the coding questions in Part 4: now asking not just, “Why am I interested in *that*?” but, “What is this a *sort of*.” Academic family is a sort of family background.

Sorting nodes

A tree structure, like any catalogue, is a logical organization of nodes, not a model of what you think you are finding out from your data. Thus each node has a place with other concepts of that sort. Treat the trees as expressing where each concept logically *fits*: use the search processes to ask questions about the *relations* between concepts. For example, if you were coding who takes responsibility for certain tasks, you would code the person and the task in separate trees. Conceptually, the person is a “who,” and the task is a “what”. The temptation is to use trees as a shortcut to represent relationships between categories, so that, for example, one might create a node called “worry about future income,” perhaps with subcategories of father and/or mother. But father also worries about his children's behaviour, and the future of the education system. And teachers also worry about the future of the education system, or the behaviour of children in it, while the politicians are confident that the education system is “in good shape” and blame parents for the poor behaviour of their children. If the person is always a subcategory of the issue, the result is a proliferation of nodes. It also makes for an inability to find a node for coding, or to find out who worried about what, or what fathers thought more generally, or how anyone viewed children's behaviour.

Instead, organize the who, what and how in separate trees, and if particular text is about many of these, code it many times (one for each of the who, what and how). Use the search tool to discover who does what how, under which circumstances. Assumptions about relationships between categories are then a matter of investigation, not an artefact of the coding system.

There are several steps to this process of creating some conceptual order in a coding system. The first critical step is a thinking-sorting process, to decide how your nodes might be arranged. Then nodes are shifted so their order reflects conceptual relationships. Normally, these steps are engaged in several cycles. But first, it would be helpful to start by simply printing a list of the nodes you have already created. Periodically throughout your project, you will want to print a list of nodes to assess how ideas are growing, what's missing, where there are duplications and whether they are forming conceptual groups. As your nodes are moved into Trees, the list of nodes becomes a practical tool to assist comprehensive and speedy coding.

Listing nodes



- From the Nodes menu at the top of the Project Pad, choose List all Nodes, asking for Titles and Addresses Only.

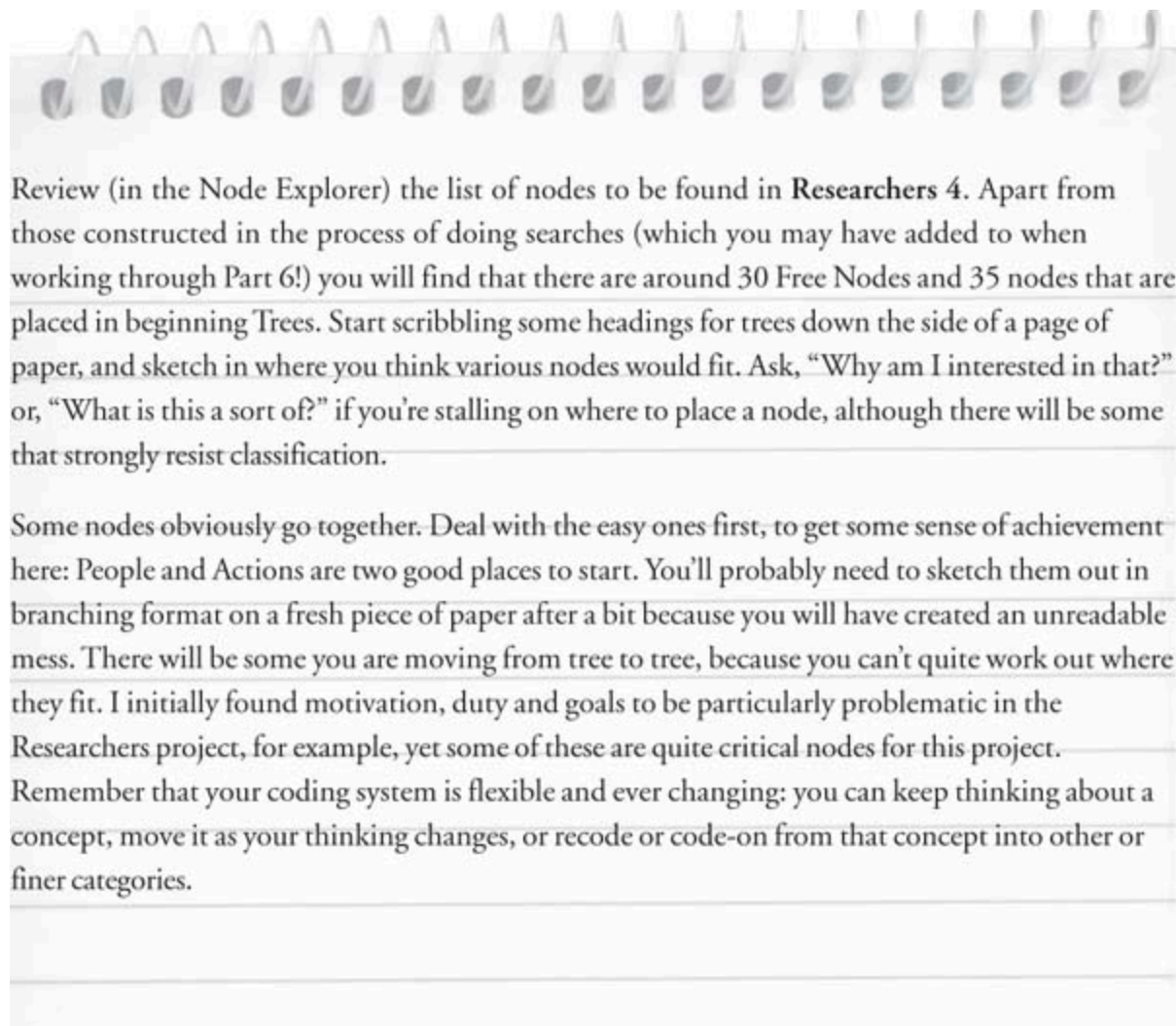
NVivo will create a report listing all nodes—first the Free Nodes, then Trees, with the trees in numeric sequence based on their node addresses. This can be edited (e.g. to place some blank lines between Trees, or to bold the parent nodes), saved and/or printed.

Grouping nodes

- Look carefully at your list of nodes, and think about them in terms of conceptual groupings. Are there a few that are sorts of a more general concept? Sometimes, those general groups will be obvious from the research design: Ask “What am I asking?” and see what groups of categories the answer contains. If it is a study of how different people, at different sites, act in different ways; those are three sorts of categories. Sometimes the groups appear during the sorting of categories emerging from the data. There may be nodes for different experiences or interpretations, such as different images of research, and grouping these becomes sensible as their variety is established.
- Sketch out the groups you are coming up with on a piece of paper, or, if you want to work in NVivo, draw them in the modeler. Try to see where each Free Node might fit (and any Tree Nodes you might already have). If there are some nodes “left over,” don’t worry—they can stay as Free Nodes for the time being (or forever!).

Most projects end up with around 10 major groupings, or trees—sometimes less and rarely much more.

- In the process of deciding on how to combine or sort your nodes, regularly browse their content, jumping from one to another, checking back to documents to see why the text that you “knew” was about something doesn’t seem to be coded there, and so on. This adds another way of seeing your text anew. Additionally, thinking about possible ways of organizing the structure of your knowledge about your data prompts rethinking about what you are looking for in that data—about the goals of your project. The questions you are seeking to answer will, of course, have a major impact on how your coding is organized.



Review (in the Node Explorer) the list of nodes to be found in **Researchers 4**. Apart from those constructed in the process of doing searches (which you may have added to when working through Part 6!) you will find that there are around 30 Free Nodes and 35 nodes that are placed in beginning Trees. Start scribbling some headings for trees down the side of a page of paper, and sketch in where you think various nodes would fit. Ask, “Why am I interested in that?” or, “What is this a sort of?” if you’re stalling on where to place a node, although there will be some that strongly resist classification.

Some nodes obviously go together. Deal with the easy ones first, to get some sense of achievement here: People and Actions are two good places to start. You’ll probably need to sketch them out in branching format on a fresh piece of paper after a bit because you will have created an unreadable mess. There will be some you are moving from tree to tree, because you can’t quite work out where they fit. I initially found motivation, duty and goals to be particularly problematic in the Researchers project, for example, yet some of these are quite critical nodes for this project. Remember that your coding system is flexible and ever changing: you can keep thinking about a concept, move it as your thinking changes, or recode or code-on from that concept into other or finer categories.

From “sorts” to structure

The “emergence” of a conceptual shape to the project is an exciting stage. You created Free Nodes up out of the data. Now you are creating a more abstract framework up out of those early categories. Working up from the particular to the general requires new nodes for the higher level, or “parent” nodes—some also referred to as “top-level” nodes, to indicate their tree position. Earlier, generally, you created nodes while coding, but not all nodes will have coding. Some now serve, rather, as “coat hangers” for ordering other nodes.

Creating Tree Nodes

You can create nodes without coding in the Node Explorer or the Coder, placing the node where you want it—indeed, you have probably already done so. These are convenient methods if you are creating an occasional node; but a third method, working in the Create Nodes dialogue, is faster if you have several nodes to create.

As you create Tree Nodes, each is automatically assigned a number. Every node is identified by a node address, a numeric code for where it is located within the index system. Addresses appear in the right pane

of the Node Explorer, in the status bar at the bottom of the Node Explorer when you select a node, and the Properties box. Most usefully, they are included in a list of nodes (as seen above). Whereas in NUD*IST 4 knowing the node address was necessary for adding coding, in NVivo you can manage quite well without ever thinking about node addresses, but they can make coding faster and are also a useful way of seeing order in your coding system.

Some trees may need branches before they can take specific nodes. For example, if nodes are in groups for setting, period of time, and the prevailing atmosphere, these could be placed as three branches of a context tree. Specific settings or times or atmospheres are then coded at sub-nodes under these branches.

Create tree nodes



- Working in the Node Explorer or the Coder, use the very intuitive method of selecting Trees, or the node at which you want to create a branch, and from the right mouse context menu selecting Create.

Alternatively

- From the Project Pad, select Make a Project Node, or from the Nodes menu at the top of the Project Pad, choose Create Nodes.
- Click on the Trees tab. (Free Nodes can also be created this way.) Any Tree Nodes you have created already will show in the left pane of the dialogue.
- Click in the Title slot on the right side of the dialogue, and type in a new tree node title. Press Enter on your keyboard (or click Create with the mouse). The node will be created.
- Type the next title (no need to click in the slot again), and Enter.
- Type and Enter until you have all the parent (top-level) nodes you want.

Create “children” of a Tree node

- In the left pane of the Create Nodes dialogue, double-click on the Tree Node for which you wish to create sub categories (“children”). Its title appears in the slot above (immediately under Tree Nodes:).
- Click in the Title slot on the right side of the dialogue, and type in a title for the first “child” node. Make sure the Address slot (under the title slot) is set at 1. Press Enter.
- Now, as before, type in a title followed by pressing Enter for as many times as you need to create all nodes at this level (rarely more than 2-4). If the node you want already exists as a Free Node, then don't create it here—it can be moved directly across from the Free Nodes area, with its coding references (text stored there) intact (see below).

Shifting Free Nodes into Trees

You have already discovered that nodes can be Cut or Copied and Pasted in NVivo, using either the Node Explorer or the Coder. The context menu on the right mouse button makes these very accessible tools also. Working in the Coder or using the context menu is fine if you want to shift just one or two while you are working on a document or node, but the best place to move nodes for the current task will be in the Node Explorer. Here you can easily extend your view of all your trees, and multiple options are offered for moving, reshaping and restructuring nodes. These options will now be presented in detail (including a brief repetition of the basic ones presented in Part 5) so you have before you the full set of choices available. It then becomes a matter of choosing the method that suits your personal style for the specific task in hand!

Remember there is no requirement that all Free Nodes be moved into Trees. It is likely that some Free Nodes will never be located under a more generic category. You may not be sure whether you want to keep that node, or perhaps it is an “outlier,” a maverick idea that will continue outside the more orderly system. For some methodologies, these unattached categories may be the crucial theoretical sparks that will inform the ultimate analysis. So Free Nodes are special. Move them into Trees only when this ordering does them justice.



You might be about to do something drastic. How long since you saved the project? Save now and also Save Project As with a different name (*before* embarking on a reorganization!). You can always return to the backup if you make a move you regret later!

When you are moving your nodes, have beside you a sketch of how you wish to organize them. A separate checklist of Free Nodes (to mark off when they have been moved) may also be helpful. But before you start on moving, it is useful to understand more fully the variety of ways in which the Node Explorer can be put to work.

Node Explorer revisited



- Click once on Free (or Trees) in the left hand pane of the Node Explorer. All Free (Tree) Nodes appear in the right hand pane, along with information about how many passages they code, and when they were created and last modified. For Tree Nodes, the specific tree address is given as additional information.
- Double-click on Free (or Trees) in the left pane. It will expand to show any Free (Tree) Nodes. They will also be listed on the right pane (as before). In the list on the left side, those which have coding at them will be shown in bold.
- Note the different icons: Tree Nodes with children have “legs” hanging from their blue icons (in both left and right panes). Those with children also have a + next to them (in the left pane) to indicate that

they can be expanded further.

- Click once on a Tree Node with children. The children will show in the right pane of the Node Explorer. Double-click and the Tree will be expanded in the left pane.
- Click on any node to see its coding status and description (lower right pane).

The two sides of the Node Explorer allow different things to be done to nodes. In the left pane:

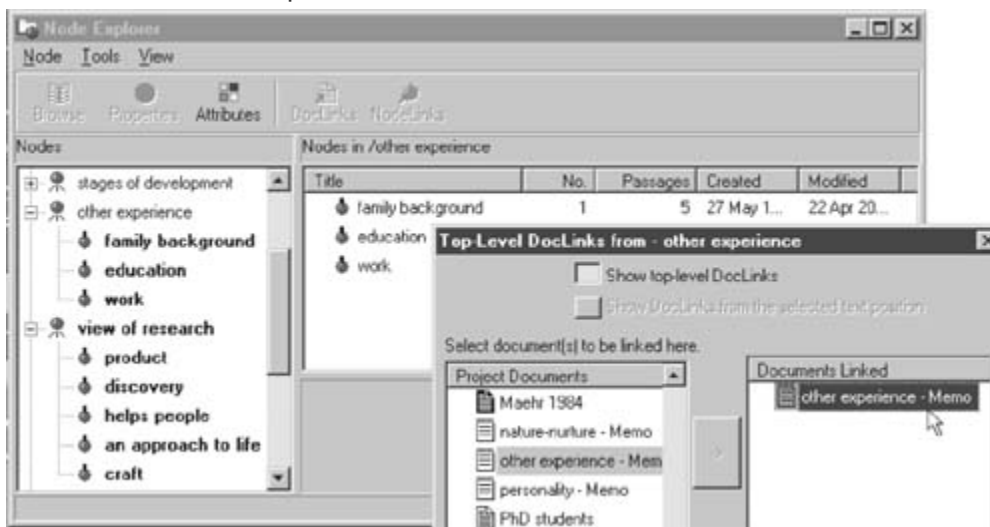
- Single click on a highlighted node title to edit its title.
- Click on any node title to select it, then on Browse to open it in a Node Browser.

In the right pane:

- Double click on any node title to open the Node Browser for that node (but this will expand the source list in the left pane—if it wasn't already—leaving the right pane blank).
- Click on any node title to select it, then click on Browse, to open the Node Browser for that node without altering the list in either pane. (This is very useful when you want to check the contents of a Free Node while you have the trees showing in the left pane).
- Select multiple nodes using Ctrl-click (e.g. if you want to move a number of nodes to the same location, or to delete multiple nodes).

In either pane:

- Select any node and click on the appropriate toolbar icon (or choose from the context menu) to access its Properties, Attributes, DocLinks or NodeLinks.



Moving Free Nodes to Trees using drag-and-drop

To “drag” a node, like text in a word processor, involves selecting the node using the left mouse button, then clicking on the selected node and moving the mouse while continuing to hold down the left mouse button, until the destination is reached.

- Expand or extend the Node Explorer (drag the bottom right corner down) so that it shows more nodes at one time.
- In the left pane, expand the Trees to show the top level nodes, and any main branches. Click once on Free, so the Free Nodes show in the right pane.
- Drag a Free Node from the right pane to the required tree (or branch) in the left pane. It will be attached to that tree or branch as a child node.
- If several nodes are to become children in the same tree, multiple select them (use Ctrl-click) and drag them together to that tree.



When you drag a Free Node to a Tree, it is copied rather than moved (rather like moving a file from one disk to another). You should then delete it from the Free Node list (see below for how), to avoid later confusion.

Moving nodes from tree to tree with drag-and-drop

- Drag from the right pane to the left, or drag within the left pane—whichever is more convenient. (This will depend on how much of the tree structure you can see in the left pane).

If you drag a node from one tree to another, it is moved rather than copied and therefore does not need to be deleted.



Sibling nodes can be rearranged by changing the nodes' addresses via the Node Properties dialogue.

Delete excess copies of nodes

- Highlight the node(s) you wish to delete (e.g. copies of Free Nodes you have just moved into a tree).
- From the context menu, select Delete Node. Confirm that you wish to delete the specific nodes you had selected—this is a useful safety check and it is worth paying attention to what is listed for deletion (can you hear the voice of experience speaking here?!).

Moving nodes using Cut and Paste

Cut and Paste is the method we recommended initially (in Part 5) if you needed to move a node. The advantage of this method is twofold: if you cut a node then paste it, you will not have to backtrack to delete it, and, you can paste the node more than once. Thus, if a Free Node comprises two separate concepts, you can paste it into two trees and develop both copies in directions appropriate to that tree. A Free Node remains on your clipboard no matter how many times you paste it, until you cut or copy another node. A further advantage of using Cut and Paste is its precision. Drag and drop must be used with caution, since it is easy to “drop” a node in the wrong place. Cut and Paste is recommended for the nervous researcher, Copy

and Paste is great for the excessively nervous!

Combining and splitting nodes

By now you should also be already familiar with merging two nodes. Merging is a strong way of solidifying a node system—without losing any coding work. As with cutting and pasting, where the original node encompasses two meanings, you may need to merge it in two places, or paste it to one and merge with another. On some rare occasions it may be necessary to recode its contents into more than one place, using the same techniques as for coding-on, i.e. to code from the Node Browser into new or other existing nodes.



Do not merge coding in this way unless you are sure: it is very difficult to undo the merge (even if you have copied the source node) because some text may have been coded at both nodes, while other text will have been coded at just one or the other.

Just checking!

Now that you have shifted all those nodes around, print out a fresh list of your nodes (Nodes menu, List all Nodes).

This will allow you to check that everything has ended up where you wanted it to be. More importantly, it will give you an overview of the conceptual shape of your coding system.



Create top-level Tree Nodes as needed for the plan you have developed from the nodes in the **Researchers 4** database. Then move Free Nodes in under them. Some will need to be moved to or merged with nodes under trees that have been already created—and maybe also preserved for attaching or merging in a second place.

When you've completed as much moving as you are happy with, you might like to see what I ended up doing. The restructured tree is there for you to see in **Researchers 5**. You will find some of the thinking behind it recorded in the Project Journal. While I did the main changes over a morning, I kept "tweaking" it every time I worked with a new document or reviewed data.

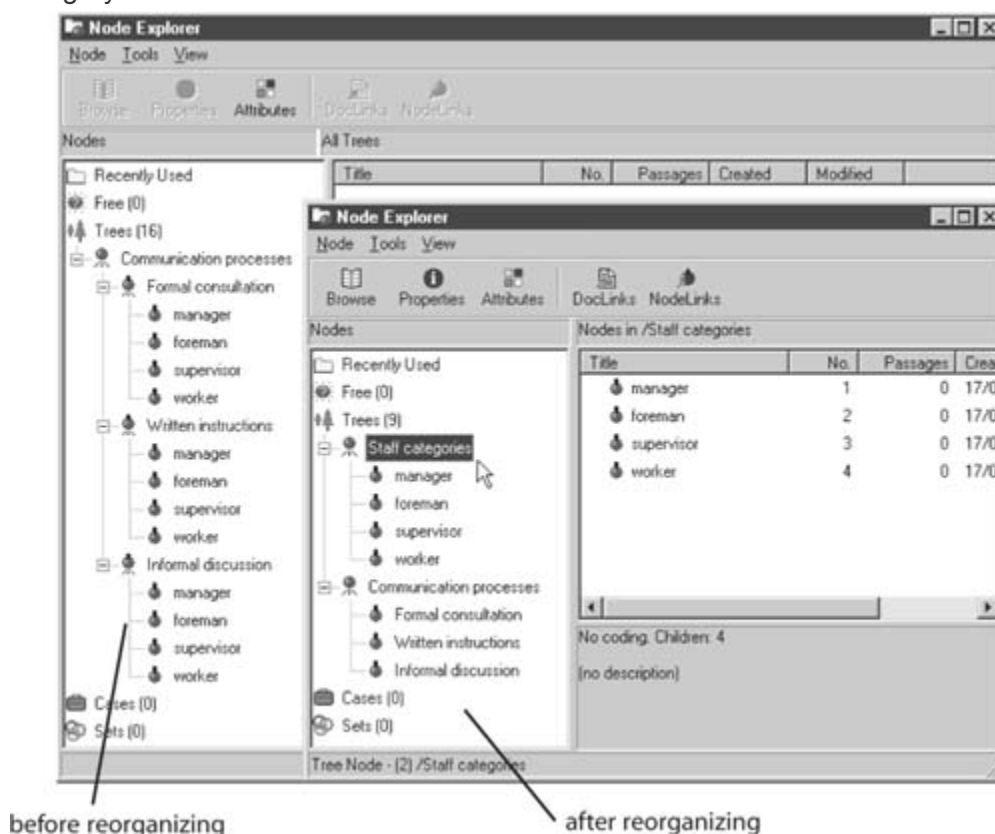
Rearranging an index tree

Most of us need to rearrange trees at some stage (read of Pat's experience below!). The most common challenge is to tidy multiple repetitions of a particular group of nodes. In the example shown, from a study

of workplace communication, the categories of people doing it may be repeated as subnodes of each type of communication. For just three sorts of communication, with four categories of staff, there are already 16 nodes. As the study progresses, the nodes will proliferate. But more importantly, the study is not keeping like concepts together (there is no one node, for example, where we can see everything the manager is doing). Unlike concepts are combined, and this will limit our ability to develop concepts. For example, subnodes of formal communication can't be developed as we see different ways of formally communicating.

Clearly the project needs two trees, one for staff categories and one for modes of communication.

Compare the Explorers below. Even at this early stage there is evident advantage in the simpler system—only 9 nodes, rather than 16, and the trees clarify what the project is about—patterns of communication by staff category.



Transformation of this index system took three simple steps.

First, the nodes that were at the top level (Communication processes) are given all coding that was previously at the subnodes (so Formal consultation now codes everything previously coded at the staff member doing it, and so on). This requires a Union search. Now the Communication processes nodes, with all their coding, can be placed in their own tree.

Then all coding that was previously at each lot of duplicated nodes (those for Staff categories) is merged (so one node now codes everything previously coded at three nodes, for example, Manager). One tree is made for Staff categories.

All you have to do is delete the original nodes (having carefully checked you relocated the

coding!)

The tools for this transformation are already familiar—gather coding at several nodes by Boolean Union search, copy, paste and merge nodes. The result is a clear framework for the study, and a change in your coding. If text is about the manager's style of formal consultation, it is coded twice. You can now easily find all that you have about that particular communication style, or all communication by the manager, and if you want to know how formal consultation was used by the manager, a Boolean Intersect search will quickly tell you.

1. Collect all the text “below” a parent node



- Choose to Search the Project Database and then to conduct a Boolean (Union) search (as you have done several times before, by now).
- In the Choose Nodes dialogue, highlight the “parent” node for which you wish to gather coding at the “children” (and “grandchildren” if there are any), e.g. Formal consultation, in the example above. Click to Add the whole Subtree to the right pane of the dialogue, then OK to have those nodes show in the search dialogue.
- Ensure that All Documents is showing as the scope for the search, then click on Run Search.
- From the Search Completed dialogue, check Show Node in Explorer and uncheck Browse Node. The node will be shown, highlighted, in the Node Explorer. This new node now codes everything that was coded at the subnodes you specified. Note the text references have been *copied* from the original nodes (which are unchanged).
- Click once on the highlighted node, and edit its title. You can use the intended name, e.g. Formal consultation, as it is not a sibling of the original).
- Create the required parent for the new node. (Call it something slightly different, e.g. Communication Styles, since you have not yet deleted the earlier tree.) Cut the node you made by the union search, pasting it under the new parent. Your new tree is begun.
- Repeat this process for each of the nodes to be brought into this tree.

2. Combine all the text of repeated nodes

- Create and title a parent node for the next new tree (Staff Categories). Copy the first of each repeated node (e.g. Manager) and Paste under the new parent.
- Copy the next repeated node and Merge with the one you just pasted. Repeat for as many repeated nodes as there are, to combine coding of all of them into one. (*If there are many repeats of each child node*, use a Union (Boolean) search to combine the text for all reports, in one step.)

3. A final tidy-up

- In the Node Explorer, click on each of the new nodes in your Tree and just check that the coding status (lower right pane) appears to be “about right,” given the data you have now gathered together.
- Now select and Delete each of the original trees, branches or nodes that have been collected and relocated into two separate places (if you work in the right pane of the Node Explorer, you can use Ctrl-Click to multiple select them).



Be careful you don't include your new tree in the list for deletion, or, if you are deleting whole branches, that there aren't other nodes still buried within them that you want to retain! Note the method used above is the safe one, copying nodes rather than cutting them. For the brave, cut and merge will be faster—but back up first!!



When I initially started working with some data about what motivated researchers (in my first attempt to use NUD*IST—version 2!) I created a node called motivation and then created a series of nodes under that of things that had the effect of motivating academics to do research. That was fine until I began wondering what to do with these “things” if they were talked about, but in a different way—perhaps as part of the researcher's developmental experience, or simply as an outcome of their work, or even as a limitation. And so they reappeared in trees for development, and/or experience, and so on—and unfortunately not in neat repetitions. After some period of struggling with this cumbersome system, application of a flash of insight saw what had become a four page list reduced to one: have a tree for impact (stimulate, develop, maintain, reinforce, limit), with separate trees for the things that had an impact, such as people, events, activities; and others for the contexts that moderated those things and for the way people felt or otherwise responded. For any interesting segment of text, I would then code at as many of these as were relevant, and always think about applying a code for impact. The difficulty I had on the way was that re-organization of the existing nodes was complicated by their varied organisation and usage, and coupled with a sense that a lot of what was in the data had been missed because of the way I had been coding it, this led to my scrapping my first coding (and its attempted reorganization) and starting again. We're hoping you can avoid a need for such drastic action with our strategies for “early prevention”!

Using structure

At this stage it is important to ask—and keep asking—why you are making these nodes, and shaping them thus. Just as coding can become a ritualistic process, so node creation and ordering can become an end in itself. If this happens, the software is not being used to its full, and your project may come to a dead stop! Is

the shape, or order, your nodes are taking going to assist you to explore and answer questions raised in and by your data?

Using trees while coding

Even before you launch into searching questions, well organized trees provide a useful tool for ensuring the thoroughness of your coding, as you are doing it. You've stopped to code a passage because an interesting issue has been raised there. Capture that, but before you leave the passage, run an eye over your tree structure as a quick visual prompt to see if there are other nodes that are relevant. Should you also note (code) who the key players are, what the context is, how people felt or otherwise responded? Do you need reminding about trees in which you want to routinely code all otherwise coded text in order to capture a key variable (maybe a contextual variable, maybe an outcome)? Making sure that the text is coded at nodes across all relevant trees will allow searches for answers to questions to be much more effective and complete.

Working with a printed list of nodes beside you is helpful for this kind of checking when coding. Before printing the list, add some visual clues to clarify the tree structure. For example, you might insert a blank line or two between each tree or remove the tab in front of the top-level node for each tree and/or make it bold.

Be prepared, of course, to add to and delete from the nodes in your list as you do further coding, and after a while, to regenerate and reprint the list. Don't expect it will ever become a fixture, even after the major sorting effort you have just undertaken!



Having nodes in trees and working with a printed list beside you also makes for easier, faster coding, as will be seen in Part 8.

Using trees to explore with searches

Having ordered and coded in this way allows you to ask exploratory and comparative questions with ease. For example, one might ask: "What issues arise in this context?" and then perhaps, for an issue that appears interesting in that context: "How does this issue look in other, different contexts?" These kinds of questions can be investigated using the Boolean search operator Matrix Intersection with the relevant nodes.

An intersection of nodes occurs where they code the same text, thus a request to NVivo to find the intersection of two nodes will result in NVivo finding all the text that is coded at (or in technical terms, referenced by) both of those nodes. Anything code at just one or the other, or at neither, will be ignored. A matrix intersection simply does this for several subcategories of a node (or several listed nodes) in parallel: the same result could be obtained by running a separate intersection search for each pair, but it would take much longer to do. The further advantage of running the searches in parallel and building a matrix is that it enhances the sense of what is being done here—that you are wanting to examine the patterning of and perhaps draw comparisons between elements of a concept when it is viewed in relation to something else.

You *could* run a matrix intersection search where you cross tabulate all issues (say) by all contexts, to produce a full table with a number of rows and columns and many cells. This can be useful for getting an overview of the pattern of distribution of what issues arise in what contexts, but for the purpose outlined here is likely to provide information overload and lead to some confusion about what is being searched for. It is better in this situation to make meaningful comparisons a step at a time, thinking about the implications of the first step before taking the second.



It is a statistical convention to use columns for “independent variables” and rows for “dependent variables” in cross tabulations. While there is no demand to maintain this convention (and such terms may be totally inappropriate to the data you are working with), for those of us with such training, it makes sense to follow the convention wherever it does seem appropriate.

Find (and compare) the issues that arise in a particular context



- Choose to Search the Project Database. In the Search Tool, choose to run a Boolean search.
- In the Boolean Search dialogue, from the slot next to Operator: select Matrix Intersection as the specific search you want to run.
- Under Find text referenced by each of these items... select to Choose Nodes. The node(s) you select for this first slot will define the row(s) of your matrix. Highlight [Issues] in your list of nodes and choose to Add > Children. The list of specific issues you have created will be listed on the right.
- Under ...pairwise with each of these items, again Choose Nodes, this time choosing just the specific [context] you wish to explore. Node(s) chosen in this pane define the column(s) of your matrix.
- Click OK to return to the Search Tool, and specify the scope for your search.
- Run Search, and click OK in the Search Completed dialogue to be taken to a view of the Matrix results. If you followed the choice of nodes suggested, this will be presented as a single column (one context) containing multiple rows of data (one row for each of the selected issues).
- In the slot next to Display: choose a form of display for your results (probably Number of documents coded is most useful). The level of shading, along with the number shown will indicate how much coding there is in any cell of the matrix display. This will give you some sense of the distribution of (in the example here) issues that arise in this particular context.
- Choose a cell of interest (a particular issue), and from the context (right mouse) menu select Browse/Code Node. A Node Browser will open to show just the text that was coded at those two nodes, that is, for both that issue and that context.
- Browse as many cells as are of interest. If you want to print the results, a coding report can be made for any cell (if you do so, type under the node address the titles of the nodes that were intersected to

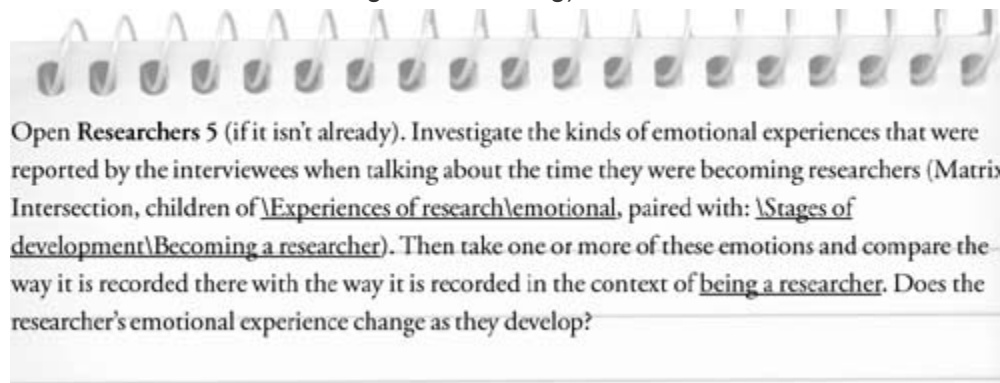
produce that result, before printing or saving it).

Compare an issue across contexts

Do you want to follow up on the previous search by taking a particular issue of interest and comparing its “shape” in different contexts. The procedure is almost identical to that outlined above.

- Run a Matrix Intersection search, as before, but this time select just the one issue node, and choose to find the text where it intersects with each of the context nodes. The result will be a table running across rather than down.
- Browse each cell of the matrix to see whether this issue takes different forms, depending on the context in which it arises.

That may be sufficient for your needs. Perhaps, however, in your reading of the text, you will become aware of specific factors operating in those contexts which need further examination or elaboration (with further searches, or additional coding and searching).



Taking stock

Research approaches use concept ordering in different ways. For some researchers, it is essential to manage all concepts in efficient index systems. For others, the ordering of concepts will be critical in identification of core categories and development of their dimensions. Projects that have short time-lines and aim at descriptive reports rather than theory construction are less likely to require very careful concept management than projects which over time aim to develop theories and validate them.

Thus the effort you spend on concept ordering will depend on your method and your goals. But no matter how important this is (indeed, perhaps particularly when it is very important) it should not become either ritualistic or onerous. Locating of nodes happens when you think about coding (What is it about and where does *that* go?). Reorganizing of nodes happens whenever you see a node that would fit better elsewhere, and this happens very frequently.

Many things are happening at once, the concepts emerging and being tested, the data building up and the ideas accruing in annotations and memos. It is easy to delay locating nodes in the heat of the chase for

meaning and understanding. Understanding however will require rigorous exploration and interrogation of just these concepts, and their orderly management will help you avoid partial or arbitrary reliance on just some concepts, or failure to see their uneven association. So the way to avoid wasting time is not to delay locating and ordering concepts, rather to build it into the process of thinking aloud about the data. Every time you pick up an idea, put it somewhere thoughtfully.

Once the tools are familiar, it is quicker to move the node than to worry about it, or make a note that later you should do so. When you move it, the coding and memo and attributes are not lost. So the profit is a constantly changing and synthesizing picture of the project and its ideas, and the price is seconds of time.

Part 8 is about saving more time by automating clerical work, so time can be better spent on the more creative work of interpretation.

<http://dx.doi.org/10.4135/9780857020079.n7>