Information Coordination:

The Management of Information Models, Systems and Organizations

Richard Veryard

First published by Prentice Hall, 1994
ISBN 0-13-099243-7

http://www.veryard.com/coordination

Chapter 4: Planning Techniques
Part b: Scoping and Clustering

4.1 Introduction

This document is one of a series of extracts from my 1994 book on Information Coordination.

This document contains the second half of Chapter 4. It describes how the information architecture (introduced in the first half) may be input to a scoping process, which uses clustering techniques to define the contents and boundaries of projects and systems.

These techniques can be applied, although with slightly different emphasis, in the different planning approaches discussed in Chapter 3.

In the hierarchical (top-down) approach, the clustering techniques are deployed on an information architecture for the entire organization, to produce a business systems architecture for the entire organization, from which detailed project plans can then be derived. In the network (organic) approach, the same techniques can be applied on a rather smaller scale, to produce coherent project scopes for individual projects and systems, each having well-structured interfaces with its neighbours.

...

4.4 Scoping and Clustering

4.4.1 Introduction

Clustering addresses the task of dividing a problem area into (semi-)independent subareas, whose solutions can be developed (in parallel or sequence) with minimum interaction effort, and minimum integration risk. Depending on your perspective and specific purpose, it can be either a method of putting small sets of objects together, or a method of dividing a large set of objects into subsets. These small sets or subsets are known as **clusters**. The word **clustering** itself can refer either to the process of defining a series of related clusters, or to the result of this process, i.e. the series of clusters itself. Each cluster can then be used to define the **scope** of something, which may be a design project, a departmental responsibility, or something else, by enumerating what it is to contain.

For information systems planning purposes, the problem area is described by an information architecture. However, we shall start by discussing some general concepts of clustering, before looking specifically at carving up an information architecture.

Some useful work on clustering in a problem-solving context was done in the early 1960s by Christopher Alexander, then at MIT. This work was taken up by Tom DeMarco, and applied to the structuring of information systems. As we have seen in the previous chapter, Alexander has largely moved away from the top-down approach to problem-solving implied by his first book, but it is a useful source for the concepts of clustering, which can be applied by top-down planners and others.

4.4.2 Project funding

…

4.4.3 Concepts of clustering

Purposes of clustering

Scoping is usually achieved by dividing the entire problem area into small objects, and then clustering the objects together to form coherent subproblems.

The purpose of clustering is to group objects together that can be worked on together. For example, to divide a large complex problem, consisting of a large number of small interconnected problems, into several groups of related problems.

Following from such a clustering, we usually construct organization structures (such as projects, departments and teams), and match the responsibilities to the clustering. We hope that at least some planning and design decisions can be taken for an individual cluster, without considering other clusters.

Another somewhat different use of clustering is to group things together for presentation purposes. For example, when displaying a large entity-relationship diagram (ERD), it is useful to establish neighbourhoods that correspond to some underlying structure or clustering. This makes user communication and documentation more effective.

Thus the clustering enables us to define the scope of a system, project, task, organizational responsibility or deliverable.

Clustering can also be used to bundle a series of information services into a commercial offering, or a series of product enhancements into a new version or release.

Criteria for good clustering

A good clustering minimizes interaction effort (during development and maintenance) and integration risk (i.e. failure to meet enterprise or technical coordination requirements).

In a good clustering, each cluster is highly coherent (i.e. it coheres, or sticks together), and there is little overlap or coupling between clusters. There is more similarity or affinity within one cluster than between two clusters.

Sometimes consistency of size is adduced as an additional criterion. In other words, each cluster should be more or less the same size. This does not affect interaction effort or integration risk. It may sometimes be convenient for the clusters to be similar in size, but it may sometimes be inconvenient. Thus if you have several project managers of equal ability, experience and aspirations, you may prefer to define several projects of the same size, based on equal-sized clusters. But if you have several project managers of different ability, experience or aspirations, you wouldn't want to give them equal sized projects. In any case, size should never be used as an excuse for incoherent or fragmented clusters.

Stability

Since the problem will undoubtedly change its shape while we are working on it, there is also an implied criterion of stability. Does the clustering remain valid, although the details change? Perhaps the clustering becomes suboptimal, in the sense that if we were starting from scratch we should prefer something else, but it may be inappropriate to change it. There should be some 'stickiness', to prevent the clustering from changing all the time. However, we do need a mechanism for abandoning a clustering if and when it is totally wrong.

Even if we don't set out to do this explicitly, the clustering will also be used to control change. Some changes (such as additional requirements) may be formulated in a way that is convenient to the current clustering. The clustering itself provides a language for identifying and formulating change. This is healthy up to a point. However, in some cases, good opportunities may be ignored or rejected (or worse, distorted so that they lose their original value), because they cannot be fitted into the framework provided by the clustering.

Extreme radicals rather welcome a periodic reclustering, turning everything over to uncover new opportunities. They will redraw diagrams, in the hope that by changing the neighbours of an object, new insights will appear. Thus they enjoy (and are likely to promote) instability. This can often be an extremely refreshing attitude, and in some organizations it is necessary. However, non-radicals hate this, and prefer things to remain in the same place wherever possible.

Thus the 'stickiness' of the clustering (how far things should change before a reclustering is appropriate) needs to provide a proper balance between the radicals and the conservatives.

Naming of clusters

"… Today we have naming of parts …"

Don't expect the clusters to correspond to existing concepts. It is precisely because we don't trust existing concepts that we have gone to all the trouble to perform clustering according to an algorithm, instead of guesswork.

Alexander even advises caution if the parts appear post hoc to resemble existing concepts. This is because people can easily be misled by the apparent continuity of the name, and ignore the formal definition of a cluster in terms of its actual contents.

> "The designer must resist the temptation to summarize the contents of the tree in terms of well-known concepts. … If he tries to do this, he denies the whole purpose of the analysis, by allowing verbal preconceptions to interfere with the pattern which the program shows him."[1]

However, sometimes a clustering will appear to group unrelated things together, and to make no sense whatsoever. Naming of clusters is always going to be difficult in such cases, if the clusters themselves don't mean anything. There are two possible explanations for such a nonsense clustering. The first explanation is that it does make sense, but not in an immediately obvious way. The true sense of each cluster may only emerge after some difficult thought, and only after this true sense has emerged can the clusters be meaningfully named. The second explanation is that the clustering really is nonsense. Errors and anomalies can sometimes result in unemployable clusters, on the 'garbage-in-garbage-out' principle. Do not be too quick to reject the clustering as nonsense, but it is always worth checking that the information used as input to the clustering is correct.

Hierarchy

In some cases, we can form clusters of clusters of clusters, thus forming a hierarchy. This clustering in stages can sometimes significantly affect the result. The lowest level clusters will be different, according to the number of stages.

Top-down or bottom-up? Do we construct the lowest-level clustering first, and then put them together to form clusters of clusters, or do we construct a broad clustering, and then divide each broad cluster into smaller subclusters? If the latter, what is the scope of the clustering? Can the clustering within broad cluster A be influenced by the contents of other broad clusters, or must it restrict itself to the contents of A?

The answer to these questions depends on the chosen coordination style. The clusterer should include everything that she has the capability of changing, depending on the results of the clustering. In a decentralized organization, there may perhaps be little point in broadening the scope of the clustering.

Note: although Alexander originally proposed a top-down clustering approach, he assumed that all the requirements have been identified, with the smallest possible decomposition. "The more specific and detailed we make the variables, the less constrained [the structure] will be by previous conceptions, and the more open to detailed and unbiased examination of its causal structure."[2] You cannot do this prior to analysis. Perhaps this means we should recluster everything between analysis and design. And I think this is correct: if you have two or three related analysis projects reaching the

[1] Christopher Alexander, Note on the synthesis of form (MIT Press, 1964) pp 127-8

[2] Christopher Alexander, Note on the synthesis of form (MIT Press, 1964) p 115

end of phase at the same time, the design area scoping should be carried out across all analysis areas, and not isolated for each analysis area.

It is usually assumed that the broad clustering is good enough to allow further subdivisions to ignore the contents of other broad clusters. But consider this: if there is a small inaccuracy introduced at each level, and if there are several levels, the accumulated inaccuracy when we get to the bottom might well be signficant. Better to have a self-correcting method.

Because the higher-level clustering necessarily ignores the lower-level of detail, we will almost certainly get different results if we redo the clustering after the detail has been determined.

This is related to the political problem of constituencies: a majority of a majority of a majority doesn't necessarily form a majority.

Heuristics and algorithms

There are several different heuristics for clustering data and activity. For each heuristic, there are different algorithms, which calculate the affinities and interactions between objects in different ways, or which may classify and weight the factors differently. Certain CASE tools include specific algorithms. (Usually each CASE tool only contains one chosen algorithm.) At present, however, we are not interested in the specific algorithms (how), but in the thinking behind the algorithm (what), which we call its heuristic.

Everyone uses the word 'heuristic' to mean something different. So this is how I'm going to use the word.

A **heuristic** defines what types of connexion shall be deemed to possess clustering force. An **algorithm** defines how to measure the clustering force of each connexion, and how specifically to combine forces. (The techniques of specific algorithms are beyond the scope of this book.)

We can distinguish between two forms of clustering: **affinity clustering**, which produces homogeneous clusters of objects of the same type, and **interaction clustering**, which produces heterogeneous clusters of objects of more than one type. For example, a heuristic to produce clusters of entity types would be an affinity clustering, whereas a heuristic to produce clusters of entity types and processes, based on the interactions between them, would be an interaction clustering.

Clustering decision

Often there are certain objects that obviously go together into clusters, and then some other objects where it is not clear whether they should belong in this or that cluster.

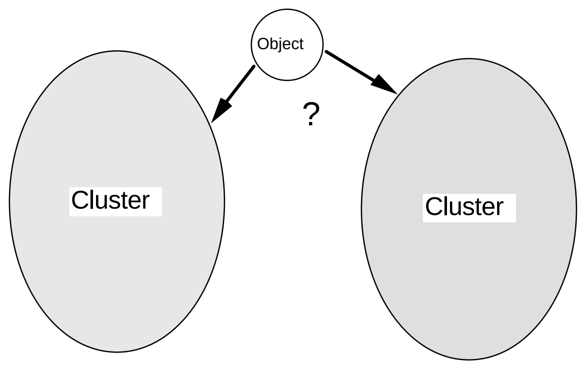

Figure 4.9 Gravitation Rule

To resolve this kind of problem, we need a 'gravitation' rule. This will often need to be provisional: we assign the object to one or other cluster, based on available information, supplemented by intuition or gut-feel, with the option to move the object later if it proves uncomfortably placed.

A heuristic is **complete** or **conclusive** if it can always resolve all clustering issues without reference to any other heuristic. In other words, every object always has one cluster that it is more strongly drawn to than any other cluster. (This isn't necessarily a good thing.) Without a single complete heuristic, we shall need several incomplete heuristics, with a **precedence rule** telling us which to apply first.

4.4.4 Types of cluster

Business areas and design areas

Much of what we have said so far can be applied at several different levels or stages. In Information Engineering, there may be a clustering at the end of Information Strategy Planning (ISP), to divide an enterprise into business areas, and a further clustering at the end of Business Area Analysis (BAA) to divide a single business area into design areas.

(In some versions of the IE methodology, the information engineer is instructed to produce design areas (aka business systems) first, and then cluster these together to produce business areas. However, although we are given criteria for this definition of business systems, we are not given explicit criteria for the second step of defining business areas.)

So what is a business area? Information Engineering material is strangely silent on this topic. We are told how to produce a business area, and what to do with it, but we are not told what its essential features are, nor even what its characteristic features are.

Data stores

A **data store** is a cluster of data objects that are to be implemented on the same hardware using the same software, and presented to the user as an integrated system. From a housekeeping point of view (e.g. such physical processes as backup and recovery), it is usually managed as a single item. However, it may sometimes be appropriate to adopt a different heuristic for scoping the logical data stores and the physical data stores. Thus the physical data stores may be physically distributed across a network, or across client/server technology, but this may be invisible to the user.

Application modules (aka systems)

An application module is a cluster of activity objects that are to be run on the same hardware, with the same underlying software, and presented to the user as a single system. As with data stores, there may be differences between the logical scoping and the physical scoping, to enable the designer to exploit client/server technology.

Business benefit packages

A benefit package is a coherent series of advantageous changes to the business operations and/or management.

A project is usually cost-justified by the expectation that it will achieve some direct or indirect business benefits. The planning and management of a project will be easier if the changes enabled by the project are a coherent set, and more difficult if the benefits are fragmented across a number of projects.

Information Engineering plans projects on the assumption that the business benefits (based on business objectives or CSFs) fall into the same natural clusters as the business functions in the information architecture. This is a useful simplification, but is not the whole truth.

4.4.5 Geo-political use of clustering

Clustering is crucial, as a technique for handling complexity. Before we consider the detail of clustering techniques for information systems, let us consider another distant example of clustering.

People seem to prefer to be grouped into political units together with similar people. This is what leads to nationalism, since people apparently want to belong to a political cluster whose affinity rule is based on nationality. This is a non-trivial problem, as demonstrated by the existence throughout history of independence movements whose fervour leads to terrorism on one side, and anti-terrorist oppression on the other side?

One of the problems of drawing political boundaries is that there will almost inevitably be minority communities left ‘on the wrong side’. (Although genocide and forced migration are often practised, under a variety of morally repulsive euphemisms, they rarely - thank God - achieve total elimination of the undesired groups.) An interesting question, that has not been properly addressed by political scientists, is whether it is better (in terms of social harmony and justice) to have small minorities (less than 10%, say), or larger minorities (more than 25%, say). It may be that a mathematical study of clustering techniques and their consequences would provide some clues.

However, since the membership of national groups is often subjective, any geo-political clustering based on nationality is likely to be unstable. Consider Yugoslavia, whose previously friendly citizens almost overnight divided themselves into hostile nation states. Religious affinities may suddenly become more relevant, as they did in Spain in 1492, and in Germany in the 1930s.

Some forms of representative democracy require a country to be divided into constituencies, each of which can elect a representative to a parliament. A constituency is a cluster. Governor Eldridge Gerry of Massachusetts invented a technique of biased clustering (known in his honour as **gerrymandering**), to ensure that the favoured party, in his case the Republicans, attained an advantage.

Note: in the first-past-the-post system, if all the constituencies were demographically identical, we should expect all the seats in a general election to be won by the same party. A random division of the country into equal-sized constituencies might well come close to this result. On the other hand, if a constituency consists only of like-minded people, it will become a ‘safe seat’, in which it is hardly worth voting. However, demographic changes necessitate periodic changes in the constituency boundaries, although there is little public clarity about the method used to redraw these boundaries (in the UK, by the Boundary Commission).

4.4.6 Alternative clustering heuristics

In this section, we shall concentrate on the clustering of data and/or activity objects.

1 Clustering by organization/management requirements

In this heuristic, data and activities are clustered according to their usage by parts of the organization, or their contribution to the management objectives of parts of the organization. This results in systems that should match organization structure. But what does this mean? The systems should provide communication within organization units, but probably not between organization units.

2 Clustering by geographical requirements

This is similar to #1, in that data and activities are clustered according to their usage by different locations of the organization. This results in systems that should match geographical structure. But what does this mean? The systems should provide communication within locations, but probably not between locations.

3 Use of existing clusters

Existing clusters (i.e. existing systems or organizations) can be used in two ways to define new clusters.

Derive clusters positively from existing clusters

Define clusters so that they replicate existing clusters. For example, activity clusters are based on existing systems; data clusters are based on existing data stores.

This will make implementation (and bridging old-new) much easier. It may be the best heuristic for short-term benefits, but is unlikely to provide a good basis for long-term improvement.

If the project boundaries more or less correspond to organizational boundaries (as with a functional organization), this probably makes intra-project coordination easier. But it may make inter-project coordination harder.

Derive clusters negatively from existing clusters

Define clusters so that they deliberately cut across existing systems and data stores. This is not a complete heuristic, and should be used in conjunction with another business-oriented heuristic. In this context, however, it may be useful to specify that, other things being equal, it may be better to break rather than respect existing boundaries.

The advantage of defining new systems and data stores that cut across existing system and data store boundaries is that it may create an opportunity to increase integration, and enhance data integrity. If the existing systems are bad enough, doing something completely different must be an improvement.

However, it requires much planning, much work, and much coordination and control. In this sense, it is a poor clustering heuristic for the short term, but may be an excellent heuristic for the long-term.

4 Hub clustering

In this approach, a single object is chosen in advance as the central or **hub** object, and the area is defined as anything connected (in a predefined way) to the hub object.

For example, the concepts of **horizon** and **inverse horizon** can be used to define clusters around an entity type. If a chain of one-to-many relationships can be constructed, all in the same direction, then a derived one-to-many relationship can be defined as their sum. The **horizon** of an entity type A is the set of entity types B_i with a one-to-many real or derived relationship to A. In other words, for each occurrence of A there is at most one occurrence of B_i. The **inverse horizon** of A is the set of entity types B_i with a many-to-one real or derived relationship to A. In other words, for each occurrence of B_i there is at most one occurrence of A.

A cluster can also be defined with an activity as the hub. The cluster around a business process may include prerequisite and controlling processes. Thus for example, authorization and error-correction processes may be implicitly included in the scope of a project, although they may use different data to the hub process, and may not have been explicitly identified when the project scope was defined.

5 Bundle clustering

In this approach, a generic type of communication is chosen as the focus of the cluster, and the area is defined as anything communicated. This may then be an information flow or **bundle.**

For example, if we analyse the communication from warehouse to factory, we may define a cluster including product data and expediting processes.

Whereas systems scoped according to clustering heuristics #1 and #2 provide for communication within an organization unit or location, bundle-clustered systems are consciously scoped to provide for communications between organization units or locations.

6 Non-binary clustering

Many clustering heuristics, such as those outlined above, assume that affinity is a binary relationship between objects. P is similar to Q, Q is very similar to R, R is slightly similar to P.

(In mathematical terms, we should want to define this as a function from pairs of objects to a quantity: F:<p,q> ∅ a.).

There are some more complicated concepts of affinity, which cannot be broken down to such a binary form. These are much more difficult to work with. Such non-binary affinities may be relevant for clustering based on data integrity, but are usually not necessary.

7 Negative clustering

Alexander is interested in positive versus negative affinities. Concurrence versus conflict. Most (if not all) of the heuristics outlined above only involve positive affinity. It seems that for clustering an information architecture or model, this is sufficient. Hoewever, for other aspects of information systems design, it may be necessary to consider both positive and negative.

8 Data clustering by data structure & integrity rules.

This heuristic clusters a data model by defining affinities based on the structure of the data model itself, with no reference to the activity model. This has the advantage that it can be carried out before the activity model is completely known. Furthermore, it is not dependent on the scope of the activity model.

Feldman and Miller[3] identified method of grouping entity types into **subject areas** based around major entity types. This is a form of hub clustering sometimes known as **dominance grouping**.

In this heuristic, entity types with relationships between them are more likely to be clustered together. Mandatory relationships will have more clustering force than optional ones. Data integrity rules that refer to two or more entity types will increase the affinity between these entity types.

[3] P. Feldman & D. Miller, "Entity Model Clustering: Structuring a data model by abstraction", Computer Journal, vol 29 no 4, 1986, pp 348-60.

For some purposes, the existence of a fully mandatory relationship between two entity types may force the inclusion of both entity types in the same cluster. This is because it can be extremely difficult to manage fully mandatory relationships that span two clusters.

Data structure clustering is complete. This is because we can rank data integrity, at least in theory, as some rules will have more strategic importance than others. Thus an entity type cannot be equally pulled towards two clusters, but one pull will prove stronger, according to the precedence rules given in box 4.5[4].

- ***Dominance grouping (around major entity types)***
- ***Abstraction grouping***
- ***Constraint grouping (data integrity)***
- ***Relationship grouping - unary***
- ***Relationship grouping - binary 1-1***
- ***Relationship grouping - binary 1-n***
- ***Relationship grouping - binary n-m***
- ***Relationship grouping - ternary and higher***

Box 4.5 Precedence rules for ER model clustering

9 Activity clustering by dependencies between activities

This heuristic clusters together activities that have to be synchronized with one another, or activities that are dependent upon one another. This makes it easier to design systems that manage the synchronicity and enforce the dependency rules. The heuristic is solely derived from the structure of the activity model, and is not dependent upon the existence or scope of any corresponding data model.

Some dependency rules have more strategic importance than others; thus this heuristic is complete.

[4] This table is taken from T.J. Teorey, G. Wei, D.L. Bolton & J.A. Kœnig “ER Model Clustering as an aid for User Communication and Documentation in Database Design” (Communications of the ACM, August 1989) pp 975-987

10 Data clustering by data use

So far, we have considered data clustering that only considers the data structure, and activity clustering that only considers the activity structure. More interesting (and arguably more useful) heuristics emerge when we consider data and activity together.

One way of clustering a data model is to start from the usage of the data model by activities. Each activity may make use of various data objects. Data objects that are used in the same way by the same activities are likely to be clustered together.

In the simplest form, a matrix of data objects against activities is populated with Xs, to indicate usage. Usually, however, different levels of usage are indicated: for example: {read, update} or {read, update, delete, create}. The latter is often referred to by its initials (CRUD), and the matrix is popularly known as a **crud matrix**.

Multiple usage (i.e. when several occurrences of the data object are used by one execution of the activity) can be regarded as more significant than single usage (i.e. when only one occurrence of the data object is used by one execution of the activity).

For many purposes, READ is regarded as less significant than UPDATE. This may sometimes be justified by the fact that a READ does not affect data integrity. However, for some purposes READ may be as important, or even more important than UPDATE. For example, a national telephone directory system, where a telephone number may need to be retrieved frequently, in many different locations, but may keep the same data for several years without modification.

This type of clustering is highly dependent on the scope of the activity model. If, for example, the activity model concentrates on the operational processes (TP), and omits decision processes (MIS), we are likely to get clusters that are suitable for operational systems, but are less suitable for MIS.

Furthermore, if we are already operating within a cluster (for example a business area), the clustering will ignore the requirements of other business areas.

Unlike data structure clustering, this heuristic is not conclusive, because there is no ranking of data usage Therefore if a data object has equal pulls towards two different clusters, this cannot be resolved by saying that one data usage is more important than another. Such open questions can only be resolved by appeal to a different clustering heuristic (or intuition).

11 Activity clustering by data used

Just as data can be clustered by analysing the similarities between usage by different activities, so activities can be clustered by analysing their similar usage of data, based on a {read, update} matrix or ‘crud’ matrix.

The same comments as #10 apply in reverse. This clustering is dependent upon the scope of the data model, and is incomplete, for the same reasons.

12 Derive data clustering from activity clustering

Having produced activity clusters (e.g. using heuristics #1, #2 or #9), it is possible to derive data clusters from the activity clusters via the data-activity usage matrix.

13 Derive activity clustering from data clustering

Having produced data clusters (e.g. using heuristics #1, #2 or #8), it is possible to derive activity clusters from the data clusters via the data-activity usage matrix.

14 Data/activity clustering by usage of data by activities

Whereas #10 and #11 are affinity clusterings, producing separate (and possibly inconsistent) clusters of data and activity, this is an interaction clustering, producing heterogenous combined clusters of both data and activity.

From a methodological standpoint, this is the most popular clustering heuristic, producing consistent clusters of data and activity, notwithstanding the fact that many tools only contain support for affinity clustering heuristics.

In the right circumstances, this heuristic produces data clusters that are reasonably cohesive under heuristic #8 and activity clusters that are reasonable cohesive under heuristic #9. This is because if two entity types are closely related in the data model, they are likely to be used by much the same processes, and if two processes are closely dependent in the activity model, they are likely to use much the same data. However, this correspondence between the different heuristics is far from guaranteed.

A particular weakness of this heuristic is that the clustering is highly dependent upon the scope of the whole model, to a far greater extent than with any other heuristic. Thus if this heuristic is used to scope design areas within a single business area, changes to the business area boundaries could have a significant effect on the design area scopes.

Data clustering by data use

Data clustering by data structure & integrity rules

Activity clustering by data used

Activity clustering by dependencies between activities

Clustering by organization/management requirements

Clustering by geographical requirements

Clustering by existing clusters.

Box 4.6 Different heuristics & algorithms for different purposes

Choice of alternatives

The preferred clustering heuristic for most purposes is the data/activity interaction clustering (#14). However, there will be situations where this is not possible, or results in trivial fragments rather than meaningful clusters.

When there is no activity model, or the confidence in the quality or completeness of the activity model is low, and a preliminary scoping is required, it may be useful to carry out a data model clustering that is not dependent on the activity model, but on the structure of the data model itself (#8), or on the mapping between the data model and some other model, such as an organizational or geographical model (e.g #1 or #2), or a breakdown of the enterprise objectives.

This will also be necessary if the purpose of a project is to design a general data repository for information retrieval, since if the activities are simply data maintenance processes (ADD, MODIFY, DELETE, VIEW, LIST), then an interaction clustering will not yield any interesting results, and a data affinity clustering will be appropriate.

With object-oriented development projects, the activities are embedded in the data objects. This again leads to uninteresting interaction clusters. A data object affinity clustering (#8) will yield better results in such situations. On the other hand, with business process reengineering projects, the data are embedded in the processes, and a process affinity clustering (#9) will be appropriate.

Hub clustering is valuable in repair projects, since it enables the extent of the changes to be reasonably predicted.

4.4.7 Alternatives to clustering

Clustering defines the scope of an area by listing the objects to be included in it. This is a complete enumeration, at least at some level of detail.

There are two main alternatives to clustering, as a technique for scoping:

1 Sometimes, an area is defined instead by a selection criterion, which is defined in advance but applied dynamically. For example, a quantified cutoff may be defined. All processes executed more than 1000 times per annum. All entity types with more than 100 occurrences. All projects with an estimated payback period of under three years. All procedures with an elapsed execution time greater than five minutes. This, which we could call **cutoff scoping**, hardly ever results in coherent or meaningful areas.

2 Intuition is frequently used, either as a technique in its own right, or as a refinement and confirmation technique.

These alternatives are not mutually exclusive. Sometimes, they may be used as supplementary technques, to resolve issues not resolved by the chosen clustering heuristic.

4.4.8 Use of clustering techniques

Scoping

An analysis or development project is scoped by a Planning study, Feasibility Study or similar exercise. To enable effective development coordination, this project scope needs to define:

- the objects of which this project is to be custodian
- the objects of which this project is to be temporary custodian (until other projects start)
- the objects that this project is to use, of which some other (existing) project is custodian

At present, we usually define project scopes in terms of high-level activities (functions or processes) and entity types (or occasionally subject areas).

Level of clustering and custodianship

Data models are usually clustered at the entity type level. However, it may sometimes be necessary or appropriate to cluster them at the subtype or even attribute level.

It is assumed that custodianship of entity subtypes, attributes, permitted values and identifiers belongs automatically with the custodianship of the entity type. There may be a few situations where we need to do something more complicated than this, but although we can probably cope with the odd exception, information resource administration would become impossible if everything was done at the smallest level of granularity.

And in any case, relationships may need to be assigned to one or other cluster.

Relationship custodianship

So what about the custodianship of relationships (or associations between subject areas)?

- If two entity types have the same custodian, then any relationships between them automatically have the same custodian.
- Relationships between entity types with two different custodians will always have one of these two as custodian, never a third. (I.e. you cannot be the custodian of a relationship unless you are the custodian of at least one entity type.)
- Custodianship of a mutually exclusive relationship rule belongs with the entity type that is subject to this data integrity.

But we need something more explicit. Here are some possible criteria.

- Which end is optional?
- Which end is many?
- Which end is used as an identifier?
- Which end locks the relationship?
- Which end does ASSOCIATE, DISASSOCIATE, TRANSFER?

There seems to be an implied concept, which optionality, cardinality and record-locking are all pointing towards. But what exactly is this concept?

Our aim is to make the systems (and therefore the projects developing them) as independent of one another as possible. This aim is supported by the following guidelines:

- A relationship across project/system boundaries must not be mandatory at both ends
- A relationship across project/system boundaries should preferably be reference-only at one end (i.e. not locking).
- If a relationship is used as an identifier, it should be reference-only at the other end.

- A relationship that needs to be updated by more than one business area should be regarded with great suspicion. (From a coordination point of view, this is worse than an entity type that needs to be updated by more than one business area.)

- If a relationship across project/system boundaries is many-to-many, the custodianship of the relationship implies custodianship of any intersection entity type that may be introduced to resolve the many-to-many.

- All relationships included in a mutually exclusive rule should have the same custodian.

However, some of these guidelines may be impractical, or incompatible with the performance and data integrity aims of the reference/update concept.

If these ideas hold water, we get the following implications:

- We need to consider strategic relationships explicitly, when defining project scopes

- Clustering algorithms should take relationships into account

- CASE tools should provide more support (including matrix support) for the analysis of relationships

- A clear logical concept needs to be defined, which may or may not correspond to the physical concept of record locking, to be analysed during planning/analysis and used to support scoping decisions.

- When additional object types are introduced into the data model, we need to have custodianship guidelines for them.

Iteration

Clustering is carried out on a model, usually represented in the form of a matrix. This model will be imperfect. Do not expect to complete the model first, and then carry out the clustering. Instead, have a first attempt at the clustering as soon as the model is 80% complete. This will highlight the areas where the model needs to be more detailed or accurate.

For example, there will often be one or two objects apparently linked with all other objects. These act as 'hinges'. For example, object **D** in the following matrix.

	A	B	C	D	E	F	G	H	I
Alpha				x				x	x
Beta				x			x	x	
Gamma				x				x	
Delta				x	x	x			
Epsilon				x	x				
Zeta		x	x	x					
Eta		x		x					
Theta	x			x					
Iota	x			x					

Figure 4.10 - Example of interaction matrix with hinge object

We could ignore **D**, and the remaining objects would fall naturally into clusters, but **D** would then remain as a hinge, coupling the clusters together. The alternative is to divide **D** itself into two or more objects. This would make the clustering simpler, but at the cost of destroying or at least impairing the coherence of **D**.

Hinge objects are usually highly generalized. Typical examples are such entity types as TRANSACTION or ACCOUNT, JOB or BUDGET, PERSON or LOCATION. So it is reasonable to ask what gives **D** itself its coherence.

Why is it necessary or advantageous for **D** to be integrated into a single generalized object, instead of divided into several more specific objects? There are two main arguments. First, there may be some business benefit that can only be achieved through this integration. Most writers cite the same example of such a business benefit: the bringing together of CUSTOMER across all products and services, instead of maintaining a series of separate customer files. Second, there may be an opportunity to share costs, through reusability.

Such arguments need to be spelled out. Some technologists believe that it is always necessary to integrate wherever possible, and claim that even if no specific benefits of integration have been qualitatively identified, let alone quantified, there is a ‘strategic’ benefit of integration.

But even if there is a benefit in having such a hinge, design the hinges to fit the door, not the door to fit the hinges.

4.5 Summary of Chapter

In this chapter, we have discussed the development, maintenance and use of strategic information models, known as information architectures, for the purposes of information systems planning. We have seen how the scopes of projects, systems and data stores can be derived from these architectures, using techniques such as clustering.

These techniques can be applied, although with slightly different emphasis, in the different planning approaches discussed in the previous chapter.

In the hierarchical (top-down) approach, the clustering techniques are deployed on an information architecture for the entire organization, to produce a business systems architecture for the entire organization, from which detailed project plans can then be derived. In the network (organic) approach, the same techniques can be applied on a rather smaller scale, to produce coherent project scopes for individual projects and systems, each having well-structured interfaces with its neighbours.

In the following chapters, we shall see how the system development projects, scoped using these techniques, can progress in parallel with well-managed inter-project interactions, and how the systems and data stores themselves, also scoped using these techniques, can be implemented and operated as integrated yet independent modules. The better the planning and scoping, the easier will this coordination be.