PATTERN LANGUAGE: A FRAMEWORK FOR LEARNING

Alan Jessop
Durham Business School
Mil Hill Lane
Durham
DH1 3LB
UK

tel: 0191 374 1233
fax: 0191 374 3748
e-mail: a.t.jessop@durham.ac.uk

PATTERN LANGUAGE: A FRAMEWORK FOR LEARNING

Abstract

Pattern languages were devised for use in architecture and urban design as ways of providing useful access to design knowledge both for the expert designer and for the lay user. They have subsequently found use in the software community for technical and organisational issues. The paper reports some applications of pattern languages (1) as a vehicle for making explicit knowledge that would otherwise remain tacit; (2) as an aid to teaching; (3) as a means of presenting statistical methods to managers.

Keywords knowledge-based systems
pattern language
learning

PATTERN LANGUAGE: A FRAMEWORK FOR LEARNING

1. Introduction

There exists an apparently paradoxical situation in which, on the one hand, it is proposed that organisations should be learning organisations, and, on the other, that individual creativity should be prized and encouraged. Whatever organisational learning might mean it surely requires two things: first, that individuals learn from experience and, second, that the resultant lessons and behaviours to which they lead exhibit some commonality or style or culture. To be creative would seem to imply doing something quite different and not necessarily to be restricted by the constraints of a common style. Exemplars of best practice, or, at least, of what works, may come only from observation of those objects or practices which exist or have existed in the world, filtered via some criteria or viewpoint.

One way to reconcile these apparently conflicting requirements is to take from practice the form of a solution but to leave open for the individual how that form is articulated in a particular application. By this means it is hoped that the given form ensures an outcome which is at least acceptable while there remain sufficient degrees of freedom for a creative attempt at a locally optimal outcome. This scheme is in common use, in musical improvisation, for instance, in which the chord sequence of a tune is used as the basis for an improvisation which will resemble the original to some extent but will be a piece of music in its own right. Similarly, we may wish to design a car which is unique and to a degree unlike others and yet is still a car. Or we may wish to design an organisation fit for a given purpose where this fitness is assured by modelling the organisations on others yet adapted for our particular use.

Pattern Language (PL) is proposed as a way of supporting this activity. A PL is a web of patterns, each of which is a simple description of

a problem and a suggestion for its solution and contains links to other patterns in the language. Although the applications of interest here are primarily managerial some of the background of PL in physical design will be helpful. This is given in the next section. It is followed by a description of four applications of the PL methodology in which the author has been involved. In the final section, what was learned from these applications is discussed.

2. Design

The design methods movement of the sixties wished to help designers, mainly architects, by suggesting formal methods of analysis and synthesis. From this activity came, eventually, the idea of the pattern language.

In learning from existing artefacts or systems what is abstracted from practice is not a solution as such but rather the form of good solutions. Within that form there is some freedom to choose to make a particular specification (Figure 1). The implementation of this model for physical design problems dates back to 1947 and the work of Zwicky on morphological analysis (Zwicky, 1967). First, an analysis of existing objects yields a list of *features*, which it is supposed any form must have. Second, for each feature a list is made of the *means* which may be employed to achieve the goal implied by the feature. Selecting one of the means listed for each feature gives a realisation which may then be evaluated. Making these selections at random is a good way of testing one's views by reaction but would prove a tedious way of exploring the many possibilities in the solution space and so it will be desirable to limit this search by noting incompatible combinations, the strategy employed in the Analysis of Interconnected Decision Areas, AIDA (Harary, Jessop, Luckman, and Stringer, 1965; Luckman, 1967; Friend and Jessop, 1969). This morphological approach provides a context for the development of PL by the architect and mathematician Christopher Alexander.

Following his work on mathematical clustering as a means of hierarchical problem decomposition and reaggregation of the solutions to the constituents (Alexander, 1963, 1964; Chermayeff and Alexander, 1966) Alexander soon began to have doubts about the realism of the assumptions necessary for his method, notably that in the decomposition the clusters were largely independent: that actions or decisions in one did not effect the others. In rejecting the hierarchical approach (Alexander, 1966) he realised that it was still necessary to provide some aid for the designers to overcome the cognitive limits which frustrate the understanding and design of large systems, such as buildings or towns. Alexander's new approach was to adopt more directly a morphological analysis, which is to say that he looked for examples of existing solutions to sub-problems rather than for ways to define those problems. It was then the task to try to record these examples of good practice in a way which was readily accessible by others. He called the result a Pattern Language (Alexander, 1975, 1977, 1979). In his pattern book (Alexander, 1977) he gives 253 patterns giving help on matters from the shape of city regions to making a front door bench. There are several points to note in relation to the intent of the enterprise. First, patterns are resolutions of conflicts, an idea familiar to designers: "the art of designing is the art of reconciliation" Archer (1965). Second, that the object was essentially what we would now call empowerment. While Alexander clearly felt this as a democratic urge he also saw it as inevitable because the complexity of the task is beyond the compass of the designer alone, indeed the very idea of the grand designer seems untenable in such situations. By using PL the complexity of the design problem, at first so daunting and intractable (because incomprehensible), is turned to advantage..

The software community discovered Alexander and PL and recognised a possible solution to the problem of description and communication of code in pursuit of greater reuse and so of increased efficiency (Gabriel, 1996). Some of the patterns which resulted are to be found in Gamma, Helm, Johnson, and Vlissides (1994), Coplien and Schmidt (1995) and Vlissides, Coplien and Kerth (1996). Much of this

work is, reasonably enough, concerned with software only. But, software is written by people working in an environment in an organisation and so it is unsurprising that PLs were also written addressing the problem of setting up and running a successful software organisation. It is this organisational application which stimulated the work now described.

3. Application 1: Industrial project management

A pattern language was developed by Parker (2000, 2001). The organisation was the development arm of a chemicals company. They had some experience in project development and management and the purpose of the language was to make concrete the experience that had been gained. Decisions had to be made about who should be involved and how the process should be started; how the problem could be entered. From the software community there are several descriptions of the development of a PL, some of which are given in the citations above and also by Meszaros and Doble (2000). Coplien (2000) sets out the method used at AT&T for entering the problem area, some of the which was discussed in a correspondence reported in Parker (2000), which says, in part,

"The way we discovered (a better term than "generated") the patterns was to gather empirical data on the structure of the organisation's communication network. We did this in role-play exercises that took about three hours each. Then we distilled the data using a tool called Pasteur that creates a social network rendition of the data.

The first key is that we did this again and again and again. Eventually, you could recognise the patterns that recurred in these social network diagrams. The second key was going back to the organisations to understand why the pictures were the way they were -- but this was usually just an exercise in recollection. Those insights combined as the basis for many of the patterns we wrote."

This process is, of course, extensive but, equally, is time consuming and in Parker's case that time was not available. Instead it was decided to use an existing PL as both an introduction and a provocation to start the process. The language was that described by Coplien (1995) the object of which was to "build an organisation and guide its software development process" and it was for this emphasis on organisational issues that this particular example was chosen.

In Parker's study a group of four was assembled to develop the patterns. All four shared a technical background but had different roles within the company so that even with this small group a reasonable spread of viewpoints was given. Six weekly meetings were used. The stages of the process were: an introduction and understanding of Coplien's patterns; writing patterns; review and rewrite patterns; review process and results. Parker taped these sessions and wrote the patterns agreed at the meetings. A language of twenty nine patterns resulted, one of which is shown in Figure 2. The shape of the pattern is clear. An issue exists in a certain context. The issue is problematic because of the need to resolve competing forces. A solution is proposed to effect that resolution, the application of which results in a different and, it is to be hoped, more tractable context. Links are provided to other patterns which are of relevance and it is via these links that the user of the language explores as much or as little as is necessary for an understanding of not just the recommendation in the solution but enough of the surrounding issues for a confident application.

4. Application 2: Course development

Following Application 1 a small exercise was undertaken to find out just how easily the process could be applied. Five members of staff at the University of Durham Business School met for a couple of hours. A brief introduction was given, including a presentation of one of Coplien's patterns. The topic chosen for investigation was the introduction of a new degree programme under severe time constraints. The discussion soon

became animated, veering off in all directions as dictated by the immediate concerns of the participants. The difficulty for the facilitator (me) was not so much writing patterns but keeping track of the patterns being generated. The subject of a pattern would be agreed and some focus thereby provided but the discussion soon started to spiral again. The process therefore involved noting the points made which were relevant for the pattern under discussion and also to note related points which themselves would form the topic for other patterns. This acted to generate an expanding structure for the language while ensuring that some progress was made on the pattern in hand. The resulting pattern is shown in Figure 3. Names of other patterns which would have been developed are:

external examiner
recruiting students
working across courses
informal working together
working together electronically
designing programmes
distinctiveness & style
student expectations
motivating staff

This brief exercise confirmed that it is possible to enter the pattern designing process quickly and that a well structured discussion results.

5. Application 3: Case analysis

It was apparent from these two applications that whatever use a PL might be the necessary discussion certainly provides a good process for the articulation of what has been learned from past experience. This is exactly one of the functions of education. It is common in business schools that the analysis of case studies is used to practice analytical skills by simulating a number of different situations. In general the case describes a problematic situation and the teacher then asks questions based upon it. Asking that

students devise a PL, or at least describe some patterns, would seem to offer a useful addition to the analytical methods available. To test this eleven MBA students took part in an experiment. A few days in advance of our meeting each was given a sample PL (Coplien again) and a written case. The case concerned the merger, in reality the takeover, of one small management consultancy by another, much larger, consultancy. Like many mergers it was proving difficult to get the constituents to work together, thereby undermining the performance of both.

We met for a morning. An introduction to the PL idea was given and the students formed into three small groups and asked to make recommendations as to what action the company should take, these recommendations being in the form of patterns. As well as providing a vehicle for debriefing and discussion patterns are also of a form which could readily provide the basis of a presentation to a client, setting out concisely recommendation and justification. Figure 4 shows one of the patterns which were written.

6. Application 4: Teaching statistics

Students and managers very rarely wish to learn statistics. What they want is solutions to problems. There are many textbooks written to a traditional linear plan, encouraging the reader similarly to start at the beginning, skipping bits where this seems sensible. But many students, MBA students certainly, will know something about some of this material and will anyway be impatient for application. The same is true of managers.

Expressing statistical method as a PL offers many advantages in this situation. The web-like structure of the language permits entry at a point which the user judges closest to the problem at hand. Links to other patterns allow access to as much or as little of the other methods as the user needs, given knowledge already possessed. Figure 5 shows one

pattern from the (developing) language. This pattern is framed as the solution to a management problem but the solution is statistical. The language may contain some patterns which are purely technical (statistical in this case) and some which are concerned with practical implementation of the results of the analyses. As Stalingaros (2000) makes clear, users may assemble a set of patterns as the basis for their own bespoke language, offering solutions to those problems which they most frequently encounter or, alternatively, to assemble the framework for a solution to a particular *ad hoc* problem.

In this pattern a *Discussion* precedes the *Solution* rather than a *Rationale* following it. This seems a better structure for a pattern which is more didactic than the other applications described above.

7. Discussion

The help provided in a pattern will usually be a mix of theory and practice, the didactic and the empirical, and it is this which makes a pattern the "best guess at the way to share solutions" (Coplien and Schmidt, 1995, p.ix). The purposes here can be summarised as three questions: is making patterns a good way of exploring what we have learned?; is it a good way of making that knowledge explicit?; is it a good way of passing that knowledge to others? The answers are: yes, yes, and not proven.

Reactions from participants in all the first three applications are that making patterns is indeed a process which is easily entered and which provides a good structure for debriefing and making explicit that which might otherwise remain tacit. The way in which discussion of one pattern generates suggestions for other related patterns testifies to the vitality of the process. The two most useful provocations seemed to be having to think about the context, old and new, and the forces which are to be resolved.

Based on the work described here it is beyond doubt that the process of making a PL has been found a useful method for exposing tacit knowledge. Starting the process was easy and the process itself not difficult, though the facilitator had carefully to juggle the completion of a pattern and the suggestions for related patterns. This was made easier by the formal structure of the patterns. Making the first pattern or two is time consuming due largely to the need to learn just this discipline necessary for keeping each pattern short and to the point and postponing other discussions to the making of other patterns. The process becomes quicker but the time needed for completion and review is not trivial, but then neither should it be.

One of the reasons why the process was easy to start was the use of a pre-existing PL both to explain the idea of a PL and to some extent to act as a template. In all three cases the pattern structure provided by Coplien was taken and used with no significant modification or reservation. While this is tribute to the robustness and appeal of the design the unanswered question is what would have happened if a more open start had been made at which the shape of a pattern was left to be decided by the participants? Given the role and knowledge of the facilitator it is highly likely that the same elements would be present, for the idea of the resolution of forces and so the transformation of context are key. A more open start would have lengthened the duration of the process thereby making application in practical situations problematic. Nonetheless, the issue of whether the initial steer was too strong remains. In Parker's study Coplien's PL provided more than just the structure of patterns in general. Because of the similarity of the applications some were used as substantive prototypes.

In two of the applications the facilitator had some knowledge of the organisation, being currently or recently employed in it. While it is, of course, necessary for patterns to be written and used by managers and practitioners of the organisation these "internal groups may be too close to the process, introducing bias through the application of organisational heuristics" (Thomas, Sussman and Henderson, 2001). Having an external

facilitator would help to guard against this, almost certainly at the cost of extending the time taken. Whatever the force of these observations, participants found that the resulting patterns were a useful articulation of knowledge, method and solution.

The usefulness of a PL to others must remain an open question, for no attempt to test it was made. Parker (2000) makes the point that rather than act as templates for solutions patterns could also be used as part of the induction of new members into an organisation. This is, after all, a social process and patterns seem to be well suited to describing this side of the business. Similarly, including on the pattern writing team people from other organisations or different parts of the same organisation could act as a vehicle for sharing and integration via the social process of writing the patterns.

Patterns inhabit a place somewhere between the rigidities of IF…THEN expert systems and the more open ended of the problem structuring methods (as described in Rosenhead, 1989, for example). In describing any complex system, and a set of beliefs and solutions is certainly complex, the formal character of the language in which the description is made acts as a variety reduction strategy, for the description will never posses the richness of the reality, and so the language needs to be chosen with some care so that the richness of description possible in the language is appropriate. A Pattern Language would seem to offer, on the one hand, this richness in its structure via contexts and linkages and yet, on the other, provides a simplicity of description in the constituent patterns.

The processes required by the writing and considered use of the language enforce a beneficial reflection. Such reflection would usually involve a group of involved practitioners. As Tyre and von Hippel (1997) note: “Collaborative processes are important because no one person embodies the breadth and depth of knowledge necessary to comprehend complex organisational problems, and also because codified abstract

‘knowledge’ is seldom sufficient to deal with actual problems in organisations.” It would seem that PL can go some way towards addressing these issues by its flexibility, its mix of the simple and the complex, and its inherently collaborative nature.

Ackowledgement

Many thanks to Dave Parker for his work on developing his PL for his dissertation and for the stimulating discussions we had. Thanks too to those other MBA students who have built patterns over the last couple of years. similarly

References

Alexander, C., 1963. The determination of components for an Indian village: In Jones, J.C. and Thornley, D.G. (Eds.), Conference on Design Methods, Pergamon Press, Oxford.

Alexander, C., 1964. Notes on the Synthesis of Form, Harvard University Press, Cambridge Mass.

Alexander, C., 1966. A city is not a tree, Design 206 (February) 46–55.

Alexander, C., 1975. The Oregon Experiment, Oxford University Press, New York.

Alexander, C., 1977. A Pattern Language, Oxford University Press, New York.

Alexander, C., 1979. The Timeless Way of Building, Oxford University Press, New York.

Archer, L.B., 1965. Systematic Method for Designers, Design Council, London. (Reprinted in Cross, N. (Ed.), 1984. Developments in Design Methodology, Wiley, Chichester.)

Chermayeff, S., Alexander, C., 1966. Community and Privacy: Towards a New Architecture of Humanism, Penguin Books, Harmondsworth, Middx.

Coplien, J.O., 1995. A generative development-process language. In: Coplien, J.O., Schmidt, D.C. (Eds.). Pattern Languages of Program Design, Addison-Wesley, Reading, Mass.

Coplien, J.O., 2000.
http://www.lucent.com/minds/techjournal/summer_96/paper11/

Coplien, J.O., Schmidt, D.C. (Eds.), 1995. Pattern Languages of Program Design, Addison-Wesley, Reading, Mass.

Friend, J.K., Jessop, W.N., 1969. Local Government and Strategic Choice: An Operational Research Approach to the Process of Public Planning, Tavistock Publications, London.

Gabriel, R.P., 1996. Patterns of Software: Tales from the Software Community, Oxford University Press, New York.

Gamma, E., Helm, R., Johnson, R., Vlissides, J., 1994. Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, Reading, Mass.

Harary, F., Jessop, N., Luckman, J., Stringer, J., 1965. Analysis of interconnected decision areas: an algorithm for project development, Nature 206 118.

Luckman, J., 1967. An approach to the management of design, Operational Res. Quarterly 18 345–358.

Meszaros, G., Doble, J., 2000. A Pattern Language for Pattern Writing, http://hillside.net/patterns/Writing/patterns.html

Parker, D., 2001. The Use of Patterns in Management, Operational Research Society: OR43, Bath (conference paper).

Parker, D., 2000. The Use of Patterns in Management. University of Durham Business School (MBA dissertation).

Rosenhead, J. (Ed.), 1989. Rational Analysis for a Problematic World: Problem Structuring Methods for Complexity, Wiley, Chichester.

Stalingaros, N.A., 2000. The structure of pattern languages, Architectural Research Quarterly 4 149–161.

Thomas, J.B., Sussman, S.W., Henderson, J.C., 2001. Understanding "strategic learning": linking organizational learning, knowledge management, and sensemaking, Organizational Science 12 331–345.

Tyre, M.J., von Hippel, E. (1997) The situated nature of adaptive learning in organizations, Organization Science 8 71–83.

Vlissides, J.M., Coplien, J.O., Kerth, N.L. (Eds.), 1996. Pattern Languages of Program Design 2, Addison-Wesley, Reading, Mass.

Zwicky, F., 1967. The morphological approach to discovery, invention, research and construction, In: Zwicky, F., Wilson, A.G. (Eds.), New Methods of Thought and Procedure, Springer-Verlag, New York.

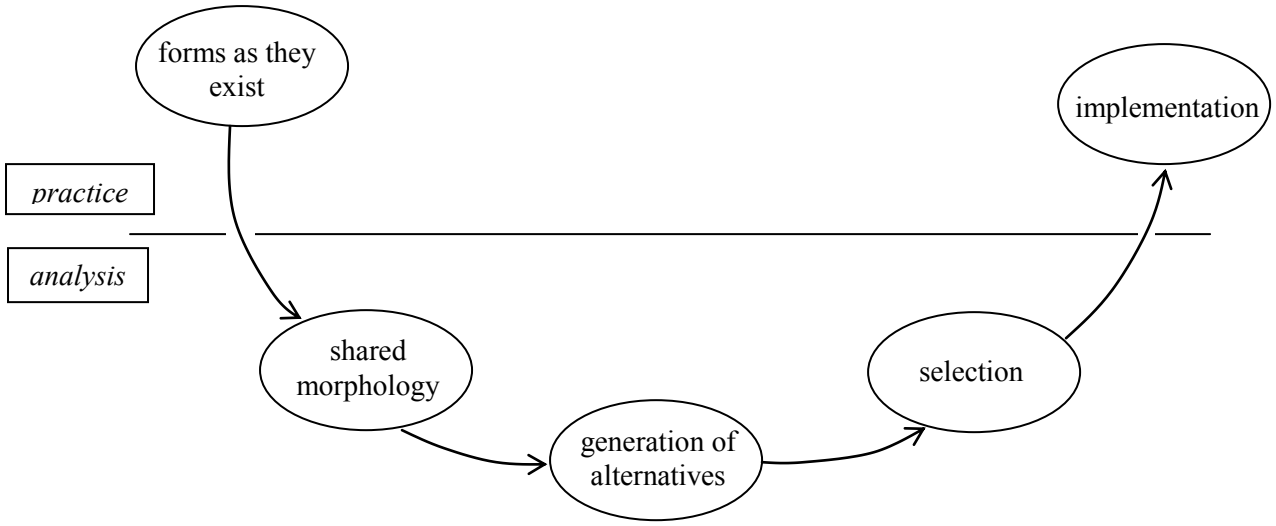

forms as they exist
implementation
practice
analysis
shared morphology
selection
generation of alternatives

Figure 1.

Pattern 32: Magic Numbers

Issue: What is a reasonable team size?

Context: The project is large enough to require decomposition into a number of specialised work units, and possibly a number of levels. [Precursor Role Clarity/Expertise?]

Forces: Too few people per team is inefficient in spreading responsibility, and for communication.

Too many people implies a scope that is too big for one person to manage.

Solution: 1. Working groups typically operate with a span of 2-4 people.

2. When applied at higher levels, the span may be up to 8 people.

3. The template can be applied to the number of active projects per team. Experience has shown that this is typically 2-3. This provides diversity without over-stretching resources.

This pattern must be used carefully in conjunction with Role Clarity [Pattern31] to ensure that teams are reasonably well decoupled, but with Integration [Pattern 35] and Gatekeepers [Pattern 62a] to moderate the information flow.

New Context: The group has a reasonable trade-off between efficiency and communication.

Background: This is learning taken from experience on projects X1 and X2.

Figure 2. Pattern for finding the size of a team (Parker, 2000)

PATTERN: STEALING COURSES

Problem Need to develop a new curriculum quickly and with limited resources.

Context A new degree has been initiated against a tight deadline. Course documentation must be prepared such that the success of the programme is assured. Taking course outlines prepared by others seems a good idea.

Forces Appropriating successful courses certainly cuts time and probably assures success (acceptance, at least). Danger of being seen as copyists. Lack of ownership.

Solution Find a similar programme offered elsewhere, preferably at an institution of at least similar status as your own. If the general purpose of this programme is close to that of the programme you are to design examine the constituent courses and copy any that meet your needs. Repeat for all potential donor programmes. Once all courses are amassed check for linkages and update if necessary. Rewrite your statement of purpose (aims, objectives, etc.).

Resulting Context This process allows for the speedy assembly of a programme. An important part is to examine the whole and make it coherent and your own. You may find some gaps and so need to design a few courses from scratch. Using this pattern you should have a programme that will easily gain recognition but not one that is distinctive or innovative.

Distinctiveness may be acquired via activities and characteristics outwith the course syllabuses; by where and how the programme is delivered, for instance.

You will have minimised the resources used. You will need to find ways to give staff ownership of the programme to encourage their commitment.

Rationale There is no need to reinvent the wheel. Courses can be taken from a number of donor programmes and assembled for your purpose provided that the contexts are similar or, if not, the differences are understood and account taken of them. This is a well used design method, though often unacknowledged.

Figure 3. Pattern for helping programme development.

PATTERN: CULTURE

Problem	Culture misfit: dominant culture too powerful
Context	The dominant culture is imposing its values on the weaker.
Forces	Both organisations have been driven by different cultures which are distinct, different and succesful in their own contexts. The imposition of the dominant culture is causing or contributing to falling profits and plummeting morale.
Solution	Take the best practices from both cultures.
Resulting Context	Both organisations will have the best elements of both cultures leading to improved work practice and happier staff.
Rationale	This will work because it recognises the distinct features of both organisations which in their own context bought comparative advantage and success.

Figure 4. Pattern derived from case study analysis.

PATTERN 6: FINDING EXCEPTIONS

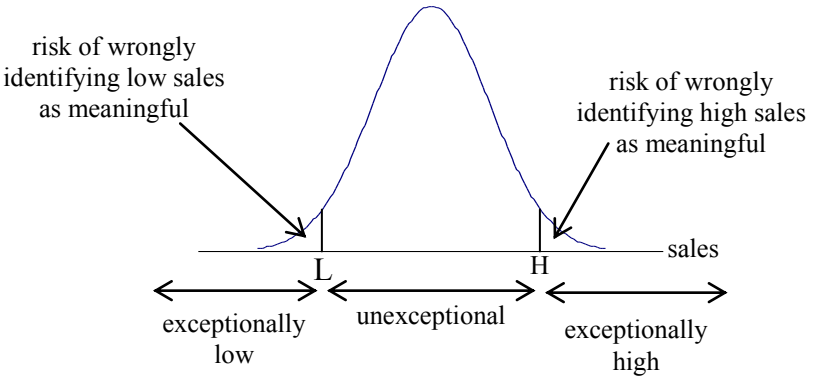
Problem How do you decide that you have found something exceptional?

Context We wish to recognise particularly good performance so that it may be rewarded or disseminated. Alternatively, particularly poor performance is to be remedied. (This is sometimes called managing by exception.)

Forces Deciding too readily that we have seen some exceptional behaviour results in unjustified action which may mean that time or money has been spent unnecessarily. Being too cautious about identifying the unusual risks missing something of significance.

Discussion An observed behaviour may be thought exceptional because it is by some distance different from the behaviour of similar things (the value of sales made a member of a sales team compared to the rest of the team). This difference could be either the result of meaningfully exceptional behaviour (good sales technique) or it could be just the lucky (or unlucky) extreme of random behaviour and signify nothing at all. In deciding which of these effects is the more plausible explanation it is usual to think of just one sort of error of classification: deciding that behaviour is meaningfully exceptional when it is not {Pattern 7: *Mistakes*}.

Solution Begin by assuming that some performance measure (sales, in this illustration) varies from person to person for reasons that have nothing to do with the individual: it could be that a client organisation just happens to land a big contract and consequently needs more from your organisation, so the sales go to whoever happens to look after this client. The distribution of sales is known and described by a probability distribution such as:



If we define some values so that sales above an upper limit, H, are said to be exceptionally high and, further, meaningfully high, then the tail probability is the risk that those high sales occurred just by chance and the attribution of meaning to them is not justified. We could similarly define some lower limit, L, to identify poor sales. We may sometimes wish to define exceptionally good and bad performance and so use both tails, or sometimes just the good or the bad, and so just use one tail.

Either we can set the limit(s) and see whether the risk is acceptable, or we could set the acceptable risk and see what the limit(s) ought to be.

Remember the difference between exceptional and meaningful. Basketball players tend to be exceptionally tall but we seek no meaningful causation for this. To set limits to identify meaningful exceptions it is always necessary that we believe there is some explanation other than randomness which might account for what we see.

Figure 5. A statistical pattern.