Attribute Focusing: Machine-assisted knowledge discovery applied to software production process control

INDERPAL BHANDARI

IBM Research
T. J. Watson Research Center
Yorktown Heights, NY 10598

Abstract.
How can people who are not trained in data analysis discover knowledge from a database of attribute-valued data ? This paper attempts to shed light on this poorly understood question by:

- presenting a machine-assisted method called Attribute Focusing for such discovery.
- describing preliminary results from the application of Attribute Focusing to improve the process of software production.

On the basis of the experience with Attribute Focusing, the paper concludes that machine-assisted discovery of knowledge by the layman should be emphasized much more than has been in the past.

1. *Introduction*

One may use the dichotomy below to characterize approaches to data exploration, data mining and knowledge discovery:

1. The automatic discovery of knowledge by a computer program.
2. The semi-automated discovery of knowledge in which a person guides a computer program to discover knowledge.

Approach 1 has received the most attention in the past. Approach 2 is beginning to attract growing attention in the face of evidence that people must be involved in the discovery loop if practical benefits are to be realized in the near term (See [14]). In keeping with the drift towards practicality, let us consider a third approach, namely, *machine-assisted knowledge discovery* in which a computer program guides a person to discover knowledge from data.

Such machine-assisted discovery does not feature prominently in the literature on knowledge discovery. On surface, there is a good reason for that. Data analysts routinely use computers to analyze data and learn from such analyses. Clearly, work on knowledge discovery must go beyond such routine use. Hence, what is the use of dwelling on machine-assisted discovery of knowledge by a person ?

That use becomes evident if one considers not the lot of the data analyst but that of the layman in this day and age. Such a person is likely to know a lot about their chosen profession but know little about data analysis. It is also increasingly likely that data pertaining to their professional activity is available in a database. Clearly, a machine-assisted method which allows them to learn more about their domain from such data should be a powerful knowledge discovery technique since it could help a lot of people improve at their jobs rapidly.

Attribute Focusing is such a method. It focuses a domain specialist, whom I shall call the *analyst*, on a small, potentially interesting part of a large amount of data in a manner that leads him to discover knowledge. The rest of this paper describes the steps that make up Attribute Focusing and early results from its application to software production process control. On the basis of that experience, I conclude that Attribute Focusing helps domain specialists to do their jobs better, and hence, make the case that machine-assisted knowledge discovery should receive more emphasis than it has in the past.

2. *Data Model and Interestingness Functions*

The data to be analyzed are assumed to be in the form of a relational table called a *data set*. The rows of the table are the records of the data and the columns are the attributes of the data. An entry (i,j) of the table specifies the value of the jth attribute for the ith record. The attribute-values

can be *discrete*, i.e., be drawn from a discrete set of values, or they can be *numeric*. The corresponding types of attributes will be referred to as discrete or numeric attributes, respectively. There are several well-known techniques to convert numeric attributes to discrete attributes by dividing the range of the numeric attribute into suitable intervals (e.g. see [17]). Hence, to simplify presentation all attributes are assumed to be discrete after noting that Attribute Focusing can apply to numeric attributes as well.

Formalizing the notion of attribute-valued data, let SS be the set of subsets of the data set that are defined by specifying a set of attributes and their values. A set of attributes may be represented by $A_1,A_2,...,A_n$ and a set of values be represented by $a_1,a_2,..., a_n$, respectively, where $1 \leq n \leq N$, N is the total number of attributes of the data set. Then, a subset of the data set that consists of records wherein the A_i's have the values a_i's may be denoted by the list $\{A_i = a_i\}$, $1 \leq i \leq n$. SSn is defined to be the set of all such lists of length n. SS is the union of SSn for all such n. To illustrate the Attribute Focusing approach, it is sufficient that we only make use of the subsets $SS1$,$SS2$, and hence, let us assume that SS comprises these subsets alone.

We are ready to describe the steps that constitute Attribute Focusing. First, automated procedures called *interestingness functions* are used to order members of SS to reflect their relative interestingness to the analyst. Let us begin by formalizing the notion of such ordering. We define the class I of interestingness functions that map some $S \subset SS$ to R^+, the set of non-negative reals. Since the relations $<, >, =$ are all defined on R^+, $i \in I$ can be used to order the elements of SS. Formally, s_1 op s_2, iff $i(s_1)$ op $i(s_2)$, where $s_1,s_2 \in S$, $op \in \{<, >, =\}$.

An application of Attribute Focusing may use one or more interestingness functions to order elements of SS. Two useful interestingness functions which are based on a degree of magnitude and a degree of association of attribute-values are described below.

I_1 is a magnitude-based interestingness function. Define:

- The probability of an attribute q having a value v, $P(q = v)$, to be the number of records in the data set in which q is equal to v, $N(q = v)$, divided by the total number of records in the data set for which q has a value, $N(q)$.
- *Choice*(q) to be the number of possible values for q.
- $I_1(q = v) = ||P(q = v) - 1/Choice(q)||$, where $||a||$ denotes the absolute value of a.

Thus, I_1 maps $SS1$ to R^+ by measuring the difference between the proportion of records that have a particular value for an attribute and the proportion of records that would have had that value had the attribute been uniformly distributed. The greater that difference the more interesting the element of $SS1$.

Next, let us define I_2, an association-based interestingness function.

- Recall, that we defined the probability of an attribute q having a value v, $P(q = v)$, to be the number of records in the data set in which q is equal to v, $N(q = v)$, divided by the total number of records in the data set for which q has a value, $N(q)$.
- Similarly, define $P(q_1 = v,q_2 = u)$, to be $N(q_1 = v, q_2 = u)$, the number of records in the data set in which $q_1 = v$ and $q_2 = v$. divided by $N(q_1, q_2)$, the total number of records in the data set for which both q_1 and q_2 have a value.
- $I_2(q_1 = v, q_2 = u) = ||P(q_1 = v,q_2 = u) - P(q_1 = v) \times P(q_2 = u)||$, where $||a||$ denotes the absolute value of a.

Thus, the probability of any attribute having a value v, is assumed to be statistically independent of the probability of any other attribute having a value u and the expected probability of their combination is computed by simply multiplying their individual probabilities. I_2 orders the elements of $SS2$ by measuring the difference between the proportion of records that have a specified pair of attribute-values and the proportion of records assuming the statistical independence of those attribute-values. The larger the difference of the observed probability of the combination with respect to its expected probability, the more interesting the combination is deemed.

In summary, the first step of Attribute Focusing involves the ordering of attribute-values based on a degree of magnitude and a degree of association of attribute-values. Such heuristics have been used before in knowledge discovery [13]. Hence, the contribution of this paper lies not in recom-

mending their use but instead in specifying where and how such heuristics be used in machine-assisted discovery.

3. *Filtering Functions*

In the second step, automated procedures called *filtering functions* prune the ordering output by an interestingness function and present the pruned ordering in a form which leads the analyst to discovery. Thus, the use of the filtering function makes viable the human analysis of a data set which contains a large number of attribute-values. Let us formalize the notion of a pruned ordering.

The class, F, of filtering functions is defined by the following model. $f \in F$ decides which of the elements of SS must be drawn to the attention of the analyst. It uses the ordering on members of SS as defined by a member of I, the interestingness functions, and basic knowledge of the limits of human processing. Essentially, f presents the top elements in the ordering in the form of no more than Y tables, each of which contains no more than X related elements, where X,Y are small integers that are decided by considering the limits of human processing. A binary function R defined on SS X SS, is used to determine whether elements of SS are pair-wise related.

X,Y and R are parameters of the model of a filtering function. We will return to how those parameters are determined in a moment. First, let us understand how an automated procedure may be set up to output tables to the analyst for a given set of parameter values.

Once X,Y and R are determined, the tables are formed by using the following procedure. Let L be the list of elements in SS arranged in decreasing order as defined by $i \in I$. Thus, the element at the top of the list has the maximum interestingness. The tables are formed by using the following procedure recursively. A table T is formed at every level of the recursion.

1. Remove top element of L and place in T.
2. Scan the list from top to bottom removing elements that are related to the top element. Place the first X related elements that are encountered in T.
3. If the Yth table has not yet been formed and L is not null, recurse, else quit.

The above procedure assumes that a single interestingness function is used to order the data. In the event that multiple interestingness functions are used, there are two possibilities. The orderings of the interestingness functions may be combined to produce one ordering, or the orderings may be distinguished and a set of tables produced for each ordering. In the latter case, Y refers to the total number of tables generated and must be partitioned between the tables generated for each ordering.

While I believe that different applications may benefit by using different values for X,Y and different instances of R, there is a guideline for defining R, and deciding the values of X and Y. Valid ranges for X and Y are computed by considering the limits of human information processing. Every table must be explained by the human analyst. Hence, X is usually restricted to around 7 entries in order to comply with the well-known fact that people are not good at retaining many items in short-term memory [12]. The limit is commonly accepted as 7 plus/minus 2.

We still need to determine Y. Since we know that a table is X entries long, we can calibrate the time an analyst will spend studying a table for a particular domain. Let that time be Z minutes. Based on the experience of working with analysts in software production, a reasonable range for Z, the time spent analyzing a single table with about 7 entries, was observed to be 5-10 minutes. I will use that range to compute Y below.

The task of the analyst is akin to that of a person taking a test in which every table represents a question that has to be answered. As reflected by the duration of college examinations, it is commonly accepted that the natural attention span for a person engaged in such a task is around 30-180 minutes. Since $5 \leq Z \leq 10$, it follows that Y should lie between 3 and 36 tables.

Now, let us turn to the binary function, R. A table being akin to a single question in a test, the items in a table should be related, i.e., they should collectively lead the analyst to discover knowledge. Since the entries of a table are elements in SS, a typical entry may be represented by a set of equalities $\{A_i = a_i\}$, where A is an attribute, a a value, and i a suitable integer. For a pair of

entries to be related, their sets should have at least one attribute in common, for then we can expect that they will lead the analyst to discover knowledge pertaining to the common attribute. Hence, R should be based on finding such commonality.

I do not wish to imply that the above method for determining X,Y,Z and R is an optimal one. On the contrary, it should be clear that more sophisticated methods could easily be devised by a human factors specialist. The only point to be made is that X,Y and Z are based on knowledge of human information processing, and R is based on the notion of related entries.

A useful filtering function, F_1 is described below. Filtering function F_1 processes two orderings. It uses the ordering produced by I_1, with X = 8, Y = 10. R is defined as follows. Recall that I_1 orders the elements of $SS1$. An element of $SS1$ may be denoted by specifying a single attribute-value such as $A = a$ since it represents the collection of records for which a attribute A has the value a. R relates elements of SS1 iff they share the same attribute.

Filtering function F_1 also uses the ordering produced by I_2, with X = 8, Y = 10. R is defined as follows. Recall that I_2 orders the elements of $SS2$. An element of $SS2$ may be denoted by specifying a pair of attribute-values such as $A_1 = a_1, A_2 = a_2$ since it represents the collection of records for which attributes A_i have the value $a_i, i \in \{1,2\}$ R relates elements of SS2 iff they have the same pair of attributes. Thus, F_1 produces a total of 20 tables each with a maximum of 8 entries, figures which are within the ranges for Y and X.

4. *Model of Interpretation*

The next step of Attribute Focusing is done manually. A model of interpretation is used to channel the thoughts of the analyst along lines which will lead to discovery. The analyst must use the meanings of the attribute-values of a table entry to relate its interestingness to the physical situation that was used to produce the data set. For instance, if the attribute-value was deemed interesting by I_1, then the analyst attempts to understand why the attribute-value had a relatively high or relatively low magnitude. If instead, the attribute-values were deemed interesting by I_2, then the analyst attempts to understand why the attribute-values had a relatively high or relatively low association.

The model of how the analyst gains such an understanding is exemplified below for a table output by F_1 which shows that $(a = v)$ is associated with $(b = u)$ but is disassociated with $(b = w)$.

- *Understand cause of interestingness:*
  - $(a = v)$ occurs frequently with $(b = u)$ but infrequently with $(b = w)$. Why ?
  - What event could have lead to such occurrence ?
- *Understand implication of interestingness:*
  - $(a = v)$ occurs frequently with $(b = u)$ but infrequently with $(b = w)$. Is that desirable ?
  - What will happen if no action is taken ?

Note that the questions above do not make reference to data analysis terms and concepts. Instead, they encourage the analyst to think directly about events in his domain. The analyst attempts to relate the entries in a table to the underlying physical situation by understanding the cause and implication of the interestingness of the entries as described above. That process ends in one of the possible ways below, after which the analyst proceeds to the next table.

1. Understanding leads to new insight that translates to action.
2. Understanding leads to new insight but no action necessary.
3. Known explanation for interestingess. No action necessary.
4. Cannot understand interestingness of a table entry. Investigate further at a later time:
   - Sample records that correspond to the table entry.
   - Study the text field (beginning of 2., "Data Model and Interestingness Functions" described the text field) to find a generalization that may explain the interestingness.

In the event that Option 4 is exercised, the analyst has more work to do, but it is focused work. He must study a sample of records from a specified subset of records.

5. *Software production process control*

Having described the steps that make up Attribute Focusing, we are ready to discuss its application to improve the process of software production. A software production process [11, 16] can be viewed as a set of activities that, if adequately defined and properly executed, lead to a high-quality software product. Defects that are discovered in the product are a sign of inadequacies in the definition or execution of the production process. If defects are to be avoided in the future, the process definition of or execution of the activities in question must be corrected. Hence, most software projects collect data on defects by describing them in writing, as well as by classifying them. Classification schemes have been used in software engineering for quite some time (e.g., see [1, 7, 8]), and most major production laboratories maintain a data base of classified data which fits the relational table model described in 2., "Data Model and Interestingness Functions." The benefit of classifying defects is that one can look for trends in the defect population as opposed to analyzing every defect individually, which can be very time consuming.

Attribute Focusing has been and is being applied to classified defect data from major software production projects in IBM laboratories in the United States, Canada and Japan to improve the process of producing software at those places. The project team plays the role of the analyst and must explain the magnitude and associations of the selected data items in a 1-2 hour meeting called the feedback session, which in turn, leads to the identification and correction of process problems. The experiences have been documented in detail [2-5]. I borrow from those results below to establish that Attribute Focusing allows software developers to improve at their jobs over and beyond current practices for such improvement. To begin with, it is instructive to examine examples of how Attribute Focusing leads to software process improvement.

TABLE 1

ATTRIBUTE-VALUE	P(attribute=value)	1/Choice(attribute)	Diff
Type = Function	33 %	13%	20%
Type = Timing	1 %	13%	-12%
Type = Condition	12 %	13%	-1%

A table similar[1] to Table 1 was output by F_1 for data from the low-level design stage of the project. Low-level design usually involves the logic specification of modules that make up a component. It succeeds high-level design which involves the partitioning of the functionality of a software product across its components [16]. Both, high-level and low-level design documents usually undergo *inspections*. During inspections [9] these documents are reviewed by people other than the author to find defects in the document. Table 1 is based on data generated by classifying defects found during low-level design inspections.

Let us understand the first entry of the table. The defect data had an attribute called *Type*, which captured how the defect was fixed. Some of its possible values are shown in the table. For instance, the value of the first entry is *function* denoting that the defects with that attribute-value were fixed by making a change to the functionality of the product. Similarly, the last entry denotes defects that were fixed by changing conditional constructs in code, and so on. For completeness, F_1 also outputs the information that I_1 used to rank the entries in the table. Thus, 33% of the defects in the data were of *type=function*, while such defects should have been 13% of the data had the distribution of *Type* been uniform. The difference is shown in the *Diff* column.

The following process problem surfaced when the team tried to explain the first entry in the table, namely, the magnitude of defects for which *type=function*. As per 4., "Model of Interpretation," the *cause* and *implication* had to be determined. On account of a weak understanding of the customer's requirements, the functionality required of a part of the product only became clear to the project team after high-level design was completed. Hence, the missing functionality had to be developed from scratch at low-level design In other words, functional aspects were being worked on for a part of the product during low-level design (*the cause*). Since functional problems should already have been addressed by the time a product enters low-level design (see definition above), this meant their process was broken, i.e., the low-level design was not doing what it should have been doing. If the requirements were poorly understood and no action was taken, the team ran the risk of similar omissions of functions surfacing later in the process (*the implication*). The

1 Exact data are proprietary information

cause could also be inferred by reading the portion of the design documents which corresponded to the part of the product which had the functional defects, and by comparing the statement of the original requirement with the functionality which was now being implemented (*corroboration* of discovery by sources independent of data).

The process problem, namely, a weak requirements stage, was thus identified and its implications considered. To correct the situation, the team decided to rework the requirements document for the project before proceeding any further. That rework was a process correction, since it was not part of the original process. A single such correction can have a major impact on the quality of the software that reaches the customer. For instance, a weak understanding of customers' requirements usually results in many major defects being found in the field. One such defect can cost a company several thousand dollars [6].

Lets consider another example. A multiple-choice questionnaire was filled out for every defect found in the field to understand how such defects could be eliminated from the next release of a major operating system. The complete experience is documented in [5]. The filtered output of I_2 was used to produce such tables as shown in part by Table 2. Every entry in a table has a pair of questions along with their associated answers. Four percentages are also specified to indicate how the entries were ranked by I_2. From left to right, they are the percentages of times the values were chosen individually followed by the percentage of times the values were chosen together followed by the percentage of times they were expected to be chosen together. For instance, the first entry in Table 2 is comprised of

1. The question *What is the best way to find similar problems* and one of its choices, namely, *Testcases*. *Testcases* is shown to be chosen in 66% of the completed questionnaires.
2. The question *Why did the problem remain hidden* and one of its choices, namely, *Low exploitation of function*. *Low exploitation of function* is chosen in 32% of the completed questionnaires.
3. *Testcases* and *Low exploitation of function* were chosen together in 27% of the cases. The expected percentage was 0.66 X 0.32 = 21%.

TABLE 2

1.	What is the best way to find similar problem ? * Testcases	Why did the problem remain hidden ? * Low exploitation of function		
	66%	32%	27%	21%
2.	What is the best way to find similar problem ? * Testcases	Why did the problem remain hidden ? * Problem not recognized		
	66%	47%	36%	31%

Let us show how a table similar[2] to Table 2 led to the identification of a process problem. The cause and implication of the interestingness (See 4., "Model of Interpretation") are indicated in parentheses. Entry 1 indicates that there is an association between defects that can be found by executing additional test cases, and the fact that such defects remain hidden because function is not adequately exercised by the testing process. That association suggests that the test development process is weak in its assessment of functional coverage (*the cause*), and hence, using the existing test process to develop more test cases will not be successful at finding similar defects (*the implication*). The discovery was corroborated by showing that the assessment of coverage by project teams improved dramatically when they used a new coverage assessment tool. That tool also represented a means to address the problem.

A summary of results from four different projects is given in Table 3. The total number of records, attributes and attribute-values in the corresponding data sets is indicated. The column *Understand Better* indicates whether the team felt they had a better understanding of their domain after the feedback session. The column *Actions* indicates whether the team took specific actions after the feedback session to rectify their process. All projects were using accepted current prac-

2 Exact data are proprietary

tices for process correction based on analysis of defect data as well. However, the last two columns of Table 3 refer only to insights which were a result of the Attribute Focusing feedback session and were not arrived at by using the other techniques.

TABLE 3

Project	Records	Attributes	Values	Understand Better	Actions
A	500	12	71	YES	YES
B	104	6	32	YES	YES
C	2602	15	123	YES	YES
D	536	6	67	YES	YES

6. *Discussion*

It is clear from the above results and examples that knowledge is truly discovered. Actions are taken to correct the process of software production based on the use of Attribute Focusing. Those problems existed prior to such use and remained uncorrected. Therefore, not only do the identified problems and corrective actions represent knowledge, they represent brand-new knowledge which was hitherto not known. The experiences also show that the mechanical part of Attribute Focusing is a crucial step in knowledge discovery. The problems existed prior to the application of Attribute Focusing but were not known to the team. The same team found the problems when presented with the output of the filtering functions. Hence, the use of the filter must be essential to the discovery. Clearly, both the mechanical and human aspects of the approach are indispensable.

Table 3 shows that the Attribute Focusing analysis helped the teams improve their process of software production over and above current practices. The cost of such improvement was of the order of person hours, that cost being incurred when the team participated in the two hour feedback session. In contrast, the cost of using other methods to correct the process is of the order of person months. The teams consisted of software developers and testers who had not received instruction in data analysis beyond what is taught in a 4-year or a 2-year college program on computer science. Hence, we see that Attribute Focusing helped domain specialists improve at their jobs rapidly, which supports my belief that such machine-assisted knowledge discovery will lead to significant practical gains since a large number of laymen can benefit from it.

Attribute Focusing has been successfully deployed to discover hitherto unknown knowledge in a real-life, commercial setting. It actually helps people do their jobs better. That kind of practical success has not been demonstrated even for advanced knowledge discovery techniques.[3]

Let us understand the difference which allows Attribute Focusing to succeed practically. There are three possible areas where Attribute Focusing may enjoy an advantage over other methods: superior mathematical algorithms, ability to process more data, the use of the analyst. Each possibility is discussed below.

An advanced knowledge discovery system such as Piatetsky-Shapiro & Matheus' Knowledge Discovery Workbench (KDW) [15] is, mathematically speaking, far more sophisticated than the interestingness functions described here. Hence, the difference cannot lie in the use of more sophisticated mathematics or theory. The relational data sets which were analyzed to provide feedback to the software teams had the characteristics in Table 3. Other discovery systems are quite capable of processing data sets of this size. Hence, the difference cannot lie in the ability to process more data. Therefore, it must lie in the use of the analyst.

Indeed, previous attempts to involve people in the discovery loop are usually presented as an integration of human and machine in which the human guides the machine to discover the knowledge in the data (see [14]). In my opinion, that does not go far enough. As evidenced by this paper,

[3] such as the collection of systems described in [14]

we must have the machine guide the human to make the discovery. Perhaps I can best make the above point by adding to text that is part of an excellent article by Frawley, Piatetsky-Shapiro and Matheus [10]. They write:

"Interactive systems will provide, perhaps, the best opportunity for discovery in the near term. In such systems, a knowledge analyst is included in the discovery loop. This approach combines the best features of human and machine: Use human judgement but rely on the machine to do search and to crunch numbers. The interactive approach requires the discovered knowledge to be presented in a human-oriented form, whether as written reports [18] or visual and sound patterns [19]."

I would add the sentence: "Or, the interactive approach requires that a model of interestingness be used to output data in a human-oriented form to a domain specialist who uses a specified model of interpretation to discover knowledge".

In summary, the Attribute Focusing approach uses an explicit model (via the use of filtering functions and model of interpretation) of the process of machine-assisted human discovery, while the other approaches, if they do use a human in the discovery loop, use a model of human-assisted machine discovery. I believe it is that difference which allows Attribute Focusing to succeed practically.

Its practical success notwithstanding, I believe that the implementation of Attribute Focusing described here is a modest beginning. Simple functions were used to implement the models presented in this paper. For instance, note that I_1 is a simple entropy-based heuristic and I_2 is a simple association-based heuristic. As observed earlier, entropy-based and association-based heuristics have been shown to be useful for automatic discovery (e.g., see [13]). Specialists in that area have made use of heuristics that are, mathematically speaking, much more sophisticated. Similarly, simple heuristics were also used to implement the Filtering functions and a simple model used for human interpretation. Clearly, specialists in the human factors and cognitive science areas could do a much better job. The present work is an initial effort based primarily on engineering intuition and common sense, and, in all likelihood, a poor reflection of what can be done with the approach by the right specialists.

In other words, man-machine approaches such as Attribute Focusing represent a means of deriving immediate and significant practical advantages by combining the results of existing research on knowledge discovery with models based on human factors and cognitive science. That potential leads me to suggest that machine-assisted human knowledge discovery should be emphasized much more than has been in the past.

6.1 FUTURE WORK

There are many directions to pursue. Information-theoretic, entropy-based measures and statistical measures of association/correlation may be used to evolve new instances of interestingness functions. For instance, when considering the analysis of numeric data, statistical correlation techniques may be used to compute coefficients of correlation, which can then be used to order attribute combinations. Similarly, new instances of filtering functions may be evolved by considering human factors issues. For instance, one can improve the usability of the tabular output of a filtering function by representing that output in a graphical form. A better understanding of the human interpretation of the output of filtering functions must be sought.

7. *Conclusion*

Let us summarize what has been learnt from this research.

- Machine-assisted discovery of knowledge by domain specialists who are not trained in data analysis should be emphasized much more than has been in the past.
- Attribute Focusing is an approach to machine-assisted knowledge discovery. It uses a model of interestingness based on magnitude of data values, association of data values and basic knowledge of the limits of human information processing capabilities, as well as a model of interpretation to guide a domain specialist to discover knowledge from attribute-valued data. It provides a useful, low-cost way for a project team to diagnose and correct their process of software production.

ACKNOWLEDGEMENTS:

Bonnie Ray for implementing the first version of the Interestingness functions; David Choi for devising a space-efficient implementation of the Interestingness and Filtering functions; Mike Halliday for implementing the relational data model and interfacing it to the interestingness and filter functions; Jarir Chaar, Ram Chillarege, Allen Dooley, Eric Hansen and George Wang for useful discussions; D. Alpert, D. Butler, C. Coppola, R. Dyer, S. Horowitz, J. Kong, M. Koury, H. Morgenstern, Y. Tan, C. Vignola, Scott Campbell, Sai Chan, Waiman Chan, Dave Knott, Tri Hoang, Dennis Koren, Ed Lekanides, Barbara Rankin, Roberto Sanchez, Beth Thurau, Beth Tobias, M. Yonezawa for participating in feedback sessions; Eric Tarver, David Brown, Janette Atkinson, Pam Jasper, and Norman Roth, for participating in feedback sessions as well as for many astute observations that have enhanced the technique; and last, but certainly not least, Anil Nigam for suggesting that I write this paper, and Glenna Young, for improving its presentation.

7.1 REFERENCES

1. Basili, V. R. and Perricone, B. T. Software Errors and Complexity: An Empirical Investigation. *Comm. of the ACM*, 27(1), 1984.
2. Bhandari, I., Halliday, M., Chaar, J., Chillarege, R., Jones, K., Atkinson, J., Lepori-Costello, C., Jasper, P., Tarver, E., Carranza-Lewis, C. and Yonezawa, M. In-process improvement through defect data interpretation. *Submitted to The IBM Systems Journal*, 1993.
3. Bhandari, I., Halliday, M., Tarver, E., Brown, D., Chaar, J. and Chillarege, R. A case study of process evolution during development. *IBM Research Report RC 18592*, December 1992. Submitted to IEEE Transactions on Software Engineering
4. Bhandari, I., Ray, B., Wong, M., Choi, D., Chillarege, R., Tarver, E., Brown, D., Chaar, J. and Halliday, M. Using Attribute Focusing to diagnose and correct the software production process. *Proceedings of the Principles of Diagnosis Workshop, Vicinity of Seattle.*, Oct. 1992. Abbreviated from IBM Research Report RC 18097, May, 1992.
5. Bhandari, I. and Roth, N. Post-process feedback with and without Attribute Focusing: A comparative evaluation. *International Conference on Software Engineering, accepted for publication*, May 1993. Also available as IBM Research Report RC 18321, Sept. 1992
6. Boehm, B. *Software Engineering Economics*, pages 39-41. Prentice-Hall, Inc., Englewood Cliffs, NJ, 1981.
7. Chillarege, R., Bhandari, I., Chaar, J., Halliday, M., Moebus, D., Ray, B. and Wong, M. Orthogonal Defect Classification - A Concept for In-process Measurement. *IEEE Transactions on Software Engineering*, 943-956, November 1992.
8. Endres, A. An Analysis of Errors and Their Causes in System Programs. *IEEE Transactions on Software Engineering*, 1(2):140-149, 1975.
9. Fagan, M. Design and code inspections to reduce errors in program development. *IBM Systems Journal*, 15(3), 1976.
10. Frawley, W., Piatetsky-Shapiro, G. and Matheus, C. *Knowledge Discovery in Databases: An Overview.* in G. Piatetsky-Shapiro and W. Frawley, *Knowledge Discovery in Databases.* AAAI Press/The MIT Press, Menlo Park, California, 1991.
11. Humphrey, W. *Managing the software process.* Addison-Wesley, Reading, MA, 1989.
12. Miller, G. The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information. *Psychological Review*, 63, 1956.
13. Mingers, J. An Empirical Comparison of Selection Measures for Decision-Tree Induction. *Machine Learning*, 3, 1989.
14. Piatetsky-Shapiro, G. and Frawley, W. *Knowledge Discovery in Databases.* AAAI Press/The MIT Press, Menlo Park, California, 1991. Edited by Shapiro and Frawley
15. Piatetsky-Shapiro, G. and Matheus, C. *Knowledge Discovery Workbench.* in G. Piatetsky-Shapiro, *Proc. of AAAI-91 Workshop on Knowledge Discovery in Databases.* AAAI Press, 1991.
16. Radice, R., Roth, N., O'Hara,Jr., A. and Ciarfella, W. A Programming Process Architecture. *IBM Systems Journal*, 24(2), 1985.
17. SAS Institute, Cary, North Carolina, SAS/STAT: User's Guide, 1990.
18. Schmitz, J., Armstrong, G. and Little, J. *CoverStory - Automated News Finding in Marketing.* in L. Volino, *DSS Transactions.* Institute of Management Sciences, Providence, Rhode Island, 1990.
19. Smith, S., Bergeron, D. and Grinstein, G. Steroephonic and Surface Sound Generation for Exploratory Data Analysis. *Proc. Spec. Interest Group on Computer and Human Interaction*, 1990.