ML Method – a basic technique of "first step analysis" to create and evaluate specifications for small database software

Bevier, U.

bevier@bussole.de

Abstract Based on the axiomatic definition of information (declared Sept. 1999) as identifiable, repeatable action and the consequences for strategy (object-action-polarisation) and structure of information processing systems as value-recording, action-transporting and targeted systems together with the physical principle of least action providing a best solution for a given problem, a triangle formation proves as most effective structure, unique in decision, easy to enhance by other triangles (components) and exactly calculable by a length closely related to the definition of the length on the information. On condition of uniformly distributed tasks it can be shown, that a hierarchical triangle structure can be optimised to reach averaged shortest distances between the interacting objects. The ML method uses this basic concept to support "first step analysis" out of words, demanding binary relations between them and resulting in a hierarchy of objects/roles, which are generally simplified to be persistent objects in the case of SOHO.

Introduction

While modern software becomes more and more challenging, modern development environments become increasingly more powerful, and allow even developers with no professional IT qualification to produce quite presentable results rather quickly. On the other hand, the advance of office automation boosts the use of computer and software by specialists.

So isolated applications spring up everywhere like mushrooms, quick acting and targeted on the needs of those specialists to support them doing their special jobs. Contrary to this simple, strictly limited origin the usefulness of these applications impresses colleagues and co-workers and so the applications often get out of control.

The ML method is made to support "first step analysis" for these specialists, primarily power users or programmers in the SOHO area, to develop applications with structures both well adapted for their problems and the needs of service and maintenance of software. The only qualification needed is to be able to describe the problem by words, the provided result is an outline of the database structure in schemes with its public methods, which well approximates the best solution under the prevailing circumstances.

The ML method and the previous 4fF method to measure databases were a result of the author's experience in nearly 15 years multifile-database-analysing, consulting and engineering (lower SAP segment). Qualified as physicist with focus on theory and mathematics, the job in the industry leads to some harsh conclusions: Because multifile-database software is a product of hundreds of PY, it can't be able to include real modern theories of software development, even if it uses modern terminology or works with "e-"s. Nowadays popular monolithic software is a result of single-case-programming managing 1000s of details with each 1000s of statements. To avoid the problem of overloaded environments hard to maintain and service and thus expensive for the clients, but to warrant the whole functionality and individual aspects of multifile databases, the author designed a database Kernel. Presenting the working prototype the real problem of software arises once again: The user must know, what (s)he wants to do. The monolithic software mostly using old environments blurs this problem by the old fashioned organisation of these software houses. That means consultants design concepts, programmers have to do it and the testers test it. Problems in concepts are therefore mostly a problem of programmers or testers or even the clients[1](!), not a problem of the consultants, but if you use a (nearly) self programming environment like a Kernel, consultants themselves must "program" and have to realise their concepts[2]. On the other hand, the same problem "analysis" arises outside the doors of the large software, which must be coupled to the modern web-environments, where analysis is a problem, too.

Searching for analysis-tools, the result was the same in highly modern environments as in the ancient ones: "First step analysis" is the job of the consultants/programmers and must be made "by experience" – in clear words: There are no tools supporting this important step of analysis.

---

[1] See [1]: "Vermutlich befragte Nucleus Anwender, denen ein Gesamtüberblick fehlt". Peter Droste, Siebel

[2] See [2] for the same conclusion by another Kernel: "Having said that, there were significant shortcomings in the final outcome that will clearly color our application of this type of solution in the future. The ease of modifying the system during the development period, when highly motivated functional representatives were integrated into the team, resulted in a level of complacency about the system's maintainability that was not warranted. Regardless of how business rules are implemented, correct analysis of the business must occur, and the appropriate translation of the rule to the data system achieved. This demands commitment, and it requires discipline."

Each existing analysis tool does only help to describe and improve the first step analysis. The author's conclusion: The problem of IT is the lack of physics. The lack of physics is the lack of mathematics, because the usual definition of information (Shannon) is a rough estimate of a technical special case with no chance to detect information and so to give clear stepwise commands for handling information, as information processing systems have to have. However, the Shannon formula is quite old, so the question itself, why nobody knows exactly, what information is, was really evidence enough for a mathematically qualified physicist to declare information. Having found the axiomatic definition, the structure and the strategy of information processing systems are quite inevitable, with the nice consequence of measurability of information processes. Thus the first step was to measure databases. In multifile-database-engineering you often must work with unknown environments, and the first step introducing a person into a new problem domain is to look at the data. This experience was bound in the 4fF-method, telling youngsters, what to count and what to look for.

But the basic problem was the problem of designing new themes. Mostly not allowed to spend time for sophisticated analysis, the consultants were forced to hurry in this vague, not verifiable phase of analysis – the only "analytical" job, which clients usually pay, is the verbal documentation of the first discussions. Actually, even in complex problems this is a good start. Putting together the concept of physical information, the structure of information processing systems and the fact, that language is a very good information processing system, so verbal documentation can be the real "first step analysis", the author developed the ML method by translating the own experience in stepwise commands, using the simplification, that each object is persistent. And because this analytical method is based on the nature of information, it can be fully programmed (dependent on the capacity of machines and memories). However, the software to support ML is really "fresh", so it is until now only useful for the SOHO domain. Professional engineers should have the useful existing analytical tools as UML and others, helping to improve first step analysis through detailed analysis, implementation and test, but enhanced with the measurements of ML journalising the states of difference to the best solution.

The first working software depending on a ML created object hierarchy is the analysis inhouse tool itself and is described as example in the book "Die Fliege" (see [4]), which is published in German only. The first sold prototype is a data-management of diesel-engines, started at zero-point out of written data, which mix up the parts lists of data and motors.

Please note: Because of discussions with friends from USA, the terminology in English has changed since the first publications, you maybe know. However, this is only a change of words, not of contents.

Basic principles and designation of terms

The existence of best solutions is provided by the axiomatic **definition of information**[3] as a group of repeatable, coherent alternations of values of an element of quality, because physical information therefore is action and has to follow the principle of least action. A length between two values of this element of quality is defined as the shortest chain of alternations of values (action) between these values.

The **ML method**[4] realises the approach to detect the best solution by constructing a hierarchical structure of triangles on condition of uniformly distributed tasks. The triangle provides uniqueness in decision and ease of enhancing by other triangles (components). Depending on the approach of the triangle structure a length is defined closely related to the definition of the length on the information, which allows a surveying, mapping and optimisation of the structure of action. The main assumption to find the course of least action is the hypothesis of shortest path, a problem like the "traveling salesman problem". This assumption was not only made by reason of least action, but by the repeatability, which is a requirement of information. **Repeatability**[3] is strictly related to well defined states. Short ways lighten the survey of the different states and therefore protect control of these states and thus the desired conservation of information during the process.

The main terms of the ML method are "item", "impulse" and "relation", distance as "length of a pair of items", "weight" and "area" as well as "scenario" and "system weighting".

"Items" are verbal symbols of parts of the problem having the essential ability to activate or accept alterations of values. In other words: they are elementary tasks, but without any details regarding to the special realisation of this task. An "impulse" is the transmission of alteration of values between items, the sending item called "need", the accepting "use". An impulse is therefore vectored, but represents an undirected relation between the items. Thus it is essential element of the communication. All needs and uses form the "neighbourhood" of an item, the "area" is defined by the product of the number of needs and uses.

A "scenario" is a collection of items of the considered problem domain including the impulses. A "thread" is a chain of impulses. "Coherence" between two items is given, if at least one chain of relations is to be found between these two items. The chain of relations with the least number of participants defines the "length" or distance between these two items. An impulse therefore fixes a length 1. Not coherent items cannot be measured

[3] See [3] for the derivation of the definition of information and consideration of limits and applicability of classical mathematics

[4] See [4] for the description and definitions of the ML method, structural requirements and definition of measurement categories

by this length. The “weight” of an item is the sum of the distances between all the rest of the items of this scenario, the “system weighting” is the sum of the weights of all items.

An active participant or role of the scenario is called an “object”. An object is an item in combination with its impulses, where it is the ending point. Therefore, an object is an elementary system of a use, bound to its needs by the impulses. In spite of the communication in a system, the object allows communication between each item (“overall communication”), independent of the direction of the impulses. In the case of simple database software a file can be deemed as persistent object, so its schema is given by the structure of the variables of an object.

However, each variable of the object, which is not a need, is only an private variable, demanding only private methods to alter values and is of no influence on the system categories.

A „system“ is a scenario of coherent items, a subsystem a subset of these items. The numerical value of the best solution, valid for small problem domains (number of items n <= 43, triangle structure, uniformly distributed tasks, w as averaged number of items in an object) is determined by experiment [4]:

$$S_+ = n*(n-1)*3 \text{ for } w <= 6 \qquad (1)$$
$$S_+ = n*(n-1)*(w/2) \text{ for } w >6$$

Procedure

The **ML method** is the approach to detect the best solution by constructing a hierarchical structure of triangles on condition of uniformly distributed tasks. It is an easy and straightforward technique to deduce a basic hierarchy of objects from a verbal description of the considered database problem, which can be used for further UML modelling. The main point of the method is a complete terminology of the problem domain along with a list of binary relations between the different terms. Depending on the approach of the triangle structure a length is defined closely related to the definition of the length on the information, which allows a surveying, mapping and optimisation of the structure of action, because it allows numerical comparisons of problems. These problems, based on pure structural features, can therefore be solved by pure structural features. Thus the framework of the relations of the terminology can be “mechanically” operated until the measurement categories approximate the optimum values.

However, the system weighting of the best value is no more than a structural feature. Yet another important factor is responsible for the best solution: the content. So the system weighting is a necessary condition for finding the best solution, but not a sufficient one. That means, that due to the content the resultant solution is to be controlled, adapted and recalculated.

Based on experience, real solutions can be approximated accurately to within 2% of the optimal solution. The advantages of these solutions become evident in the realisation, which is shortened by the better understanding provided by the procedure of the ML method and the observed reduction of conceptual redesigns.

The aimed result to put even Non-Computer-Scientists in the position to construct isolated applications expandable and easy to service, has been reached because of the fact, that only verbal capacities are demanded and the technical implementation of the algorithms are ignored, being deeply rooted to the technical options of their development environments.

Of interest to professional software developers will be the fact that the result of the ML method as a “first step analysis” is a verbally based skeletal structure for UML modelling.

Determination of items (your job)

Items can be determined from the given documents. It is good advice to create a verbal description of the problem and this should be done. In general, this is helpful in rounding-up ambiguities, disunity and conflicts in the preliminary stages of the job, providing a written document that is useful not only for communication with clients, but also for system documentation. The description should be clipped and precise, equal facts should be named by equal words.

The nouns closely related to the problem form the basic list of the items. These items should be accurately defined, if possible using the items of the list.

Determination of impulses (your job)

Impulses are vectored, binary relations between items. The first approach can be started by using the definitions of the items itself. If there are items of the list, it can be assumed, that the verbal description catches a real relationship between these items. The direction of the impulse, and therefore the definition of which item is the need and which is the use, can be derived from the following questions:

- is „item 1“ part of „item 2“? Does “item 2” therefore have “item 1” in any way?

– does “item 2” need results from “item 1”?

– does “item 2” have to control “item 1”?

– does “item 2” follow “item 1” because of anything else?

Items without any needs or uses are nonsense. So, for each useful item it must be possible to find at least one need or use.

Structural calculable measurements (wait and see)

Items: For each item, needs and uses are entirely allocated. The product of the numbers is called its area. Items with zero area show, that they are accepting input from outside the system, or they deliver output to the outside.

The total number of items stamps the sizing of the system [4]:

– 1 object for n <=7

– 2 objects for 7 < n < 13

– 1-subsystem for n <= 43

– 2 subsystems for 43 < n <= 85

Please note: we only consider small problem domains (1 subsystem, n <= 43).

The sizing provides the aspired number of objects on condition of uniformly distributed tasks, which means uniformly distributed items. The number of objects can be made by a rough estimate:

$$o = (n-1)/(w-1) \qquad (\mathbf{2})$$

with n = total number of items, w a natural number near the square root of n as averaged number of items in an object.

List of distances: To get all the distances of the items, we create the list of distances as a square matrix. The distance of an item to itself is Zero. To provide defined values, we use a default value for the other elements of the matrix: n+1.

The next step is to consider impulses: each pair of items, connected by impulses, represents the length 1. Then we look for elements of the matrix with the default value and check, if it is possible to create a chain of relations with only 2 participants between the considered pair of items. That means, that it is possible to find an impulse to a third item, which is correlated by a direct impulse to the desired second item. Then the distance between item 1 and item 2 is 2. If all pairs of length 2 are found, the next step is to find the chains with 3 participants and so on.

Then, the weight of the items is the sum of all lengths to other items of the scenario. It measures the “meaning” of the item in this field of activity by a measurement of usage and integration.

System weighting: The sum of all weights form the system weighting, which surely is far from the best solution at the beginning of the process.

Based on experience, the first value is less than the system weighting of the best solution, because in this state of first approach the items as elementary tasks are mostly unstructured and therefore too much clustered.

List of threads: Starting from the items with zero area the pervading threads can be determined, that means the chains of impulses, which transports input from outside the system to the output back to the outside.

We know, that items with zero area without needs are definitely no objects. So, the remaining items showing zero area must form the top of the hierarchy of objects, the one delivering the output back to the outside world, to our users. Therefore we arrange the objects according to the pervading threads. Equal threads or threads, being part of other ones, will be ignored.

Single object: Derived from the lists of threads and objects the single object is an item including its impulses, so the totality of its needs, shown by the threads. The needs represent the public variables and methods to alter values of these variables.

Trouble with objects

Due to the structural approach of the ML method to form an hierarchical structure of triangles on condition of uniformly distributed tasks and short distances we can find some basic object failures, which can be checked in the given solution.

Pushy items: Pushy items are objects of more than one use (output), because these objects build ambiguous threads and risk the intended clearness of states.

Fussy items: Fussy items are “relatively overloaded” items, which “have too much to do” because they have more items to control than the others (too much input, too much needs).

Iffy items: Iffy items are items with only one need and therefore indicating, that they maybe are not worth to be an object, but should be part of an object.

Trouble with the system

We can also find some basic mistakes in the system:

Spinner: Spinners are objects which are both the start and end of a chain of impulses, thus creating a cycle. This heightens action (and work!) without any reasonable result instead of lowering action to find the structure best following the principle of least action.

Facelessness: A faceless system is a system without input or output items. It's a special case of spinners, but with the real disturbing feature, that objects can not be found. Because this system does not provide both types of items with zero area, you cannot find pervading threads. In spite of this, single spinners can appear and disappear on and off during the process of analysis.

Splitting: A system with more than one output item can be either a result of spinners or a result of missing coherence of the subsystems. That means, that groups of chains of action exist without any communication between them. In fact, this structure is a structure of two independent, isolated scope of functions. If this is true, these two systems should be treated independent to lower complexity and so lower accident sensibility. But if this is not true, missing communication is evident. Communication is build of impulses, thus this problem gives evidence of missed impulses.

However, missed impulses can only be found due to the content, not by structural means.

Structural trouble shooting

Facelessness: In case of facelessness you have 3 causations: the first are small problem domains ($n < 13$), because there are not enough items to find a reasonable structure –the only suggestion is: look for another item, providing input or output services.

The next is missed output items. The structural suggestion is to get the lightest item, which has the greatest distance to some input items. For "lightness" means averaged short distances to the other elements of the system, so indicates high integration. "Far from input" shows an item, which has to manage already refined results. So the assumption is, that this maybe are enough doings, this could be the result of all the jobs of the system and therefor this item should be the one, which has to deliver the output.

The opposite and last case is the missing of items being able to accept input. The structural suggestion here is the heaviest item, far from output.

Spinner: It is not good advice to delete impulses, for this may delete necessary communication. Based on experience, most results of such attempts are splittings. It's not a good advice, too, to reorder impulses to other objects, this mostly leads to more spinners. Of interest is, that the advice is to reverse impulses, which does not affect the system weighting.

Pushy and fussy items: First step is to look for similarities in the group of needs or uses. Here you can delete impulses, because the communication does not need more than one to transport the desired action.

Without any similarity you have to rearrange items, related to the system weighting. If the actual system weighting is below the system weighting of the best solution, you know, that your system is too much clustered, so too "dense". This is an indication of redundant communication, so you need more objects to optimise division of labour, you should break down some objects. If the actual system weighting is too high, the system needs more communication, is too "loose". In general, you should delete objects by splitting the important items or break down the unimportant to simple items and subordinate them to other objects to concentrate the flow of action.

In the case of pushy or fussy items, split the lightest items, because they are the most integrated. The conclusion: The redundant communication is to be found there. On the other hand, the lightest item demonstrates importance, so it is a real object and should not be entirely dissolved. The heaviest items of the pushy or fussy items should be rearranged to simple needs of other objects.

Iffy items: Iffy items should be ignored at the beginning of the process. If you find some in the last states of progress, split the connecting impulses to other objects, so that the both parts of an iffy item are now simple needs of another related object.

Splitting: Splitting as result of pushy items is to be cleared by the elimination of pushy items. The splitting as result of distinct scope of functions cannot be solved by structural considerations because of the missing interaction between the two parts.

To clear this problem you have to create or rearrange impulses due to the content.

Completion of the structural calculable process (your job to check)

Completion of the structural process can be:

- Facelessness of small problem domains
- Splitting in case of missing communication
- cycles of suggestions, that means that the structural suggestions lead to their own initial states
- indeterminability of states, so that the automatic criterion cannot select a unique element
- close-knitted chaining of the elements, so that the complexity is above the capacity of the calculators
- there is no more trouble to be found

The last case is the best case, the one, which is assumed to be the best solution of the problem. But in fact, it is only a structural solution, so both the hierarchy of objects and the detailed structure of the single object should be checked due to the content.

As long as you find disaccords, you have to restart the procedure, check the items, check the impulses and recalculate the scenario in an iterative way, until both the structure is optimal and matches with the users view.

References

[1]: http://www.dulcian.com/papers/BR%20Symposium%202001/Berg_BR.htm, by Roland Berg, ThinkSpark and Kevin Holt, The Kernel Group
[2]: „Umfrage bringt Siebel in Bedrängnis“, Computerwoche 40/2002 v. 04.10.2002, ISSN 0170-5121, S. 1,3
[3]: „Physik der Information“, Bevier, F.F., ISBN 3-935031-03-03
[4]: „Die Fliege oder Das Handwerk der Datenbank-Programmierung“, Bevier, F.F., ISBN 3-935031-02-05
[5]: http://www.ifi.uio.no/~samvsys/comet/Comet.html by Arnor Solberg, Arne-Jørgen Berre, University of Oslo

Related Work Section

Because ML is derived from the axiomatic definition of information, which is declared in September, 1999 not by a university and not by a huge company, this definition is not very well known. Thus no related works and no more related references are very likely, because ML really supports the first step analysis by using the new concept of the physical information. As no other product, it allows everybody able to discuss a problem to construct a hierarchy of objects, which usually must be fully described and improved through detailed analysis, implementation and test by well known methods like UML, OOram or COMET for complex components (see [5]).

Appendix: Tool and Examples

The book „Die Fliege oder das Handwerk der Datenbank-Programmierung“ offers a CD with a demo version of the little inhouse tool to help calculating the needed measurements and finding the structural suggestions. The tool is written in the 4GL Omnis Studio©Version 3.0, the book itself demonstrates the method at full length, showing 3 examples. The first example is the construction of the inhouse tool (demo version) itself.