# Metrics for Dynamics : How to Improve the Behaviour of an Object Information System

Jean-Louis Cavarero
Laboratoire CNRS / I3S, 06560 Sophia - Antipolis, France
jlcava@aol.com

**Abstract.** Which is the main difference between a traditional information system (i.e. built with an entity relationship model or any non object oriented model) and an object oriented information system ?
From our point of view, the fact that in the first one the processes are not located somewhere, and in the second one the processes (operations or methods) are encapsulated in classes. The choice of the right classes to home every operation is essential for the behaviour of the system. Dynamic metrics allowing to evaluate the behaviour of a system when it runs are much more useful than any static metrics used to tell if the system is correctly built or not.
We propose in this paper, a new approach to evaluate *a priori* the behaviour of a system, by taking into account the notion of **event cost** and the notion of **time**. The final goal of this approach is to deliver information on the way operations have to be placed in classes in order to get better performances when the system is running. An optimisation tool is being under construction in order to provide solutions to this problem.

## 1 From requirements to classes

Basically the final goal of an object oriented design is to **build classes** (and the best ones as possible).
From our point of view, this building process is divided into four main steps : Step 1 : Requirements analysis ; Step 2 : Events identification ; Step 3 : Operations specifications ; Step 4 : Classes design.
Considering steps 3 and 4, the two questions are : **Which** operations have to be found out and specified to provide correct answers to each event (step 3) ? **Where** are located these operations in classes (step 4) ?
We may note that the first three steps are not specific to object oriented design. The first two ones are exactly the same and the third one, which consists in finding out the right processes, is slightly different because in one case we have to specify programms or requests and in the other we have to specify operations.
It means that the fourth step constitute the main difference between a traditional design and an object design (in traditional design processes are not located and this fourth step does not exist). It also means that to optimise the processes in a traditional design we just have to check if they are well done, in an object design we have, in addition, to check if they are well placed.

## 2 Object oriented metrics

A great lot of works have been done on metrics (6, 10), specially on static metrics. The goal is to be able to say if a system has been correctly built according to the rules implied by the model which has been used.
Most of these metrics are based on criteria such as : number of classes, number of links, number of inheritance and composition relationships, ratios attributes/operations in each class, depth of inheritance hierarchies, … and even cyclomatic number of the class diagram.
The idea, of course, is that a *well built* diagram (a good design) will induce a good code and a very efficient system, and it is true that a good design is always a key for a successfull implementation.
As example, Genero proposed in (7) a very interesting table of metrics. By applying these metrics the designer can evaluate his class diagram and is able to re-build (some parts or the whole of it) in order to guarantee that what he has built is well built (16). Unfortunelately the true finality of an information system is not to be well built, but is to give entire satisfaction when it runs, that means to give correct answers to any event coming in, and the quickest as possible. The proposal of metrics is useless if their practical use is not demonstrated (1, 2, 5, 13, 15). So their validation must be carried out from two points of view ; first the empirical validation and then the formal validation (14, 3).
Dynamic metrics appeared then, to allow an evaluation of the system behaviour (12). Most of them are based on use cases (and of course on UML concepts) from initial works of Marchesi (11) and extended by Hendersons - Sellers in (8 and 9) and Yacoub (17). Most of these metrics are metrics for requirements. They are used to assess when a requirement is too complex, at the wrong level, or too superficial (4 and 19).
But to evaluate the **system behaviour** the main notion to deal with is **time**. We have to be able to say if the system will run well over a full period of time, we have to be able to take into account what will happen during

that period and **optimise** the system performances over that period. Of course, the result of this optimisation will give a new arrangement of operations in classes (and the best one if possible).

## 3 How to evaluate the cost of an event

When an external event comes into the system, it starts the performing of all the operations involded in the process which must give the right answer. This process is shown in the **sequence diagram** of the event (Fig.1).

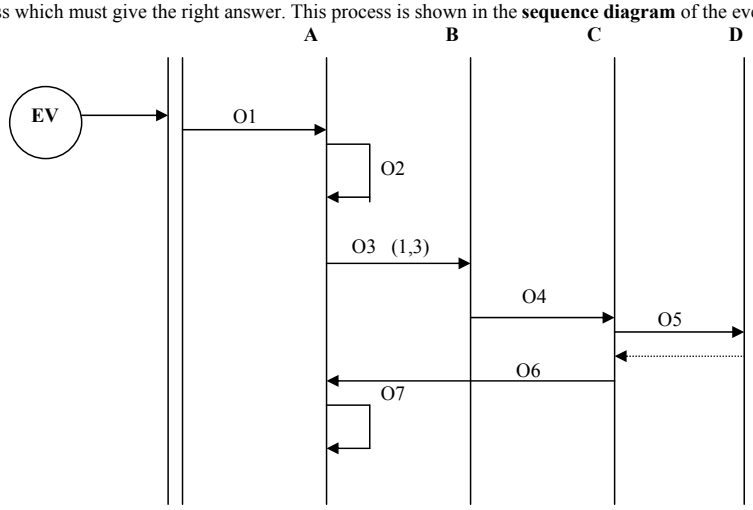


**Fig. 1.** The following sequence diagram shows that event EV first activates the operation O1 in class A, then O2 still in A, then O3 in class B (3 times), then O4 in class C which calls O5 in class D and finally O6 and O7 in class A.

The cost of each event is given by its sequence diagram : it is the summ of the costs of all the operations which are activated (that is the cost to perform the code and the cost to get another object if necessary), added to the costs of their calls (that is the cost to go from one class to another). In the example above, there are 7 operations activated (one of which three times) and only 5 calls.

When some operations are located in wrong classes the cost of the event grows up because the number of calls involved is too high, especially when there are deep inheritance or aggregation hierarchies.

We may consider that :

- the cost of the calls is more important than the cost of operations, because it is more expensive to call an operation (from one class to another one) than to perform the code associated to an operation (which is generally a very short bit of code). In C++, the cost of a call is between 2 or 3 times more important than the average cost to perform a basic operation (it is even more in JAVA).
- the cost of operations performing is not optimisable, that means even when the operations are not well located and are very expensive, the system has to perform them anyway and whereever they are (assuming that the analysis of the operations required by the event has been done correctly).

For these two reasons, the cost of the calls is the **main factor** entering in the computing of the event cost, and the only one which is optimisable.

For the event Evi (which involves Ni operations and Mi calls), the cost will be :

**$\sum$(j = 1 to Ni) Cost of Operation j + $\sum$(k = 1 to Mi) Cost of Call k**

(with Ni > = Mi because some operations are located in the same classes, the second term being very often more higher than the first one)

So the first conclusion is that to decrease the cost of an event, we must **minimize the cost of the calls by moving some operations from one class to another.**

The total cost, for all events, is given by :

**$\sum$(i = 1 to P) Cost Evi**, where P is the number of events.

## 4 How to evaluate the cost of the system

All the events have not the same importance according to their frequency. Each event has its own probability of appearing, so we have to weight each event by its probability.

The cost to run the system will then be :

**C = ∑(i = 1 to P) PRi x Cost Evi**

But the probability of an event is **depending on time**. Some events are very frequent at the beginning of the life cycle of the system and then become rare, others become more and more frequent, and so on. That means we have to consider that the probability of each event is a fonction of time : PRi(t). The figure 2 shows different types of fonctions PR (t) :

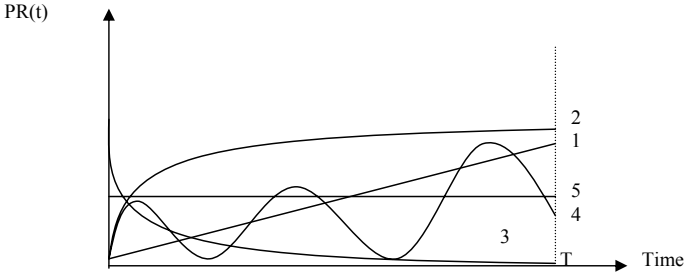


**Fig. 2.** Event 1 has a linear increasing probability (for instance *new client order*) ; Event 2 has a logarithmic increasing probability (for instance *new client registration*) ; Event 3 has a decreasing probability (for instance *new product registration*) ; Event 4 has a cyclic probality (for instance *new client order for seasons products*) ; Event 5 has a constant probability (for instance *stocks daily updating*).

Of course, knowing or estimating the function PR(t) of each event is a very difficult task, but it is definitely necessary if we want to evaluate correctly the **system behaviour** through a long period of time, because it allows to weight each event by giving more importance to the most frequent ones (and not the most complex ones which are perhaps very rare).

The final and right formula to compute the cost to run the system at the instant t is then :

**C(t) = ∑(i = 1 to P) PRi(t) x Cost Evi**

The next step is then to minimize this cost over a given period [0,T] (usually 2 or 3 years to guarantee the system stability). The **global cost GC** to minimize is given by :

$$GC = \int_0^T \sum (i = 1 \text{ to } P)\ PRi(t) \times Cost\ Evi$$

## 5 How to minimize the global cost of the system

Since the terms Cost Evi are not depending on time, this integration becomes :

$$GC = \sum (i = 1 \text{ to } P) \int_0^T PRi(t) \times Cost\ Evi$$

and then $GC = \sum (i = 1 \text{ to } P)\ Ai \times Cost\ Evi$

where **Ai** is equal to $\int_0^T PRi(t)$ which represents the weight of the probability on the period.

This global cost has to be minimized by using the only terms on which we can have an influence (the cost of operations calls), that is by modifying the **locations of operations**.

In other words, **we want to build the best class diagram** for the next 2 or 3 years, the one which will be the most efficient, in regards to all events, once coded.

The location of each operation in its class may be represented by a matrix (classes, operations), where a X means that operation i belongs to class j, (this matrix is equivalent to the class diagram, but easier to handle).

This matrix has to be optimised on the period [0,T].

Unfortunately, it is not possible to solve this problem by using a step by step method (such as learning algorithms for instance) because we have no predictive model and obviously because it would be impossible to rewrite the code at each learning step. It is neither possible to imagine an easy mathematical solution, because we have to optimise on a given period of time a function in which the terms depending on time are given.

We have to optimise the performing of the system over the chosen period in order to be able to say which is the **best matrix** on that period and then code it. This optimisation will start with the highest terms, by taking them one by one in a decreasing order. We first take the highest **Ai x Cost Evi** and minimize it by decreasing Cost Evi with a new arrangement of operations in the matrix, and so on.

## 6 Conclusion and acknowledgements

We never talked about what is happening to attributes. Of course, it was deliberate. Actually, we proposed in (20) an algorithm to provide the best arrangements « attributes/methods » in classes. This algorithm is based on the *encapsulation principle* which induce that when all the attributes an operation deals with, are in the same class, the operation is located in that class. This principle is still respected here : it means that the only operations which can't be moved (by the optimisation) are in the class of their attributes. In the opposite case, when an operation is moved, the attributes remain in their class because it won't change the number of calls.
A tool is being under construction. Some experiments have already been done from a prototype and clearly show that this approach can provide a great increase of the system performances in some cases. The most significant example came from an object information system composed of more than three thousands operations (and 117 classes) where 324 operations where removed from their initial classes, as a result of a 2 years period optimisation. The final goal of this research work is to provide the **best class diagram** as possible (and so the best code) for the future software, by taking into account what the system will have to answer to, that is all the coming in events. We wish to thank the CNEDI06 (Centre National d'Etude et de Développement des Caisses d'Allocations Familiales) for its financial support and technical help in the implementation of the optimisation tool and experiments.

## References

1. Basili V.R., Brian L. and Melo W. : A validation of object oriented design metrics as quality indicators. IEEE Transactions on Software Engineering, N°10, 1996
2.Briand L., Bunse C. and Daly J. : A controlled experiment for evaluating quality guidelinescon the maintainability of object oriented designs. IEEE Transactions on Software Engineering, 513 – 530, 2001
3.Brito e Abreu F., Zuse H., Sahraoui H. and MeloW. : Quantitatives approaches in object oriented software engineering. Object Oriented Technology : ECOOP'99 Workshop Reader, Lectures notes in Computer Sciences, Springer – Verlag, 326 – 337, 1999
4. Costello R. and Liu D. : Metrics for requirements engineering. Journal of Systems Software, 39 – 63, 1995
5. Fenton N. and Pfleeger S. : Software metrics : a rigourous approach. 2nd Edition. London, Chapman and Hall, 1997
6. Genero M. : Defining and validating metrics for conceptual models, PHD Thesis, University of Castilla La Mancha, 2002
7. Genero M., Jimenez L. and Piattini M. : A controlled experiment for validating class diagram structural complexity metrics, OOIS'02, Montpellier, 372 – 383
8. Hendersons – Sellers B. : Object oriented metrics – Measures of complexity. Prentice – Hall, Upper Saddle River, New Jersey, 1996
9. Hendersons – Sellers B., Zowghi D., Klemola T. and Parasuram S. : Sizing use cases : how to create a standard metrical approach, OOIS'02, Montpellier, 409 - 421
10. Lorenz M. and Kidd J. : Object oriented software metrics : A practical guide. Prentice – Hall, Englewood cliffs, New Jersey, 1994
11. Marchesi M. : OOA Metrics for the unified modeling language. Proceedings of the 2nd Euromicro Conference on Software Maintenance and Reengineering, 67 – 73, 1998
12. Melton A. : Software measurement. London, International Thomson Computer Press, 1996
13. Poels G. and Dedene G. : Measures for assessing dynamic complexity. 19th International Conference on Conceptual Modeling (ER2000), Salt Lake City, Lectures Notes in Computer Sciences, Springer – Verlag, 499 – 512, 2000
14. Romero J., Pastor P. and Belenguer J. : Methodological approach to software quality assurance through high level object oriented metrics, OOIS'02, Montpellier, 397 - 408
15.Schneidewind N. : Methodology for validating software metrics. IEEE Transactions of Software Engineering, 410 – 422, 1992
16. Schuette R. and Rotthowe T. : The guidelines of modeling : An approach to enhance the quality in information models. International Conf. on the E/R approach, Singapore, 1998
17. Yacoub S., Ammar H. and Robinson T. : Dynamic metrics for object oriented designs. 6th IEEE International Symposium on Software Metrics, 1998
18. Whitmire S. : Object oriented design measurement, John Wiley and Sons, 1997
19. Zowghi D., Offen R. and Nurmuliani D. : The impact of requirements volatility on the software development lifecycle. Procs International Conference on Software (ICS 2000), IFIP World Computer Conference, Beijing, China, 2000
20. Cavarero J.L. and Lecat R. : La conception orientée objet : évidence ou fatalité, Ed Ellipses, Paris, 2000