OO OR NOT OO: WHEN OBJECT-ORIENTED IS BETTER QUALITATIVE ANALYSIS AND APPLICATION TO SATELLITE IMAGE PROCESSING

Ann Gates, **Vladik Kreinovich**, **Leticia Sifuentes**, and **Scott Starks**
Department of Computer Science and
NASA Pan-American Center for Environmental and Earth Studies (PACES)
University of Texas at El Paso, El Paso, TX 79968, USA
emails {agates,vladik,leticia}@cs.utep.edu, sstarks@utep.edu

Choosing a Software Methodology: an Important Problem of Software Engineering

In software engineering, there are currently many methodologies of designing software. Each methodology has its own advantages and drawbacks, and it is therefore very important to choose an appropriate methodology for each problem. Unfortunately, this choice is often made by trial and error. For big software projects, this trial-and-error approach can be very time- and resource-consuming. It is therefore desirable to come up with reasonable recommendations that would help designers choose the optimal software design methodology.

OO or Not OO?

One of the major choices in designing a software, a software package for data processing or a database, is whether we should use an object-oriented approach. "Object-oriented" (OO, for short) is a "hot" word. This term includes many useful features, one of them being the extreme reusability. Of course, programmers have been aiming at reusability long before OO methodology was formulated: suffice it to say that reusability is one of the main reasons behind *modularity*, the idea of designing a software product as a collection of independently usable *modules*. However, only the OO methodology makes an extreme use of reusability:

- in pre-OO programming, the major requirement for a module is that it should work correctly under the conditions in which it is used in this particular software package; if the corresponding *pre-conditions* are not satisfied, the module may not necessarily work correctly;
- in OO programming, software components are designed to serve as independent objects, that must simulate, as closely as possible, real-life objects, and must, therefore, be as generally applicable as possible.

Of course, the more reusable the module, the better, so at first glance, using OO is always an advantage (and this is exactly what OO proponents say). However, as every other advantage, this additional reusability comes at a price:

- to make a module more reusable, we impose more requirements on it (by requiring that it *always* work correctly, even when the preconditions are not satisfied);
- this additional requirement makes a module more difficult to write and slower to run.

So, to make a meaningful choice of a software methodology, we must weigh its advantages (e.g., reusability) versus its potential disadvantages (e.g., increased complexity and running time).

Comment. The reader should bear in mind that although reuse is an import part of object orientation (OO), object orientation is more than unlimited reuse. In this paper, however, we will only handle the reuse aspects of OO.

Our Results and Future Work

In this paper, we present several preliminary results that will help in deciding whether we want a full reusability (as in OO) or a limited one (as in more traditional, pre-OO modular programming). These results will deal with the two major aspects of the resulting software: complexity (broadly understood) and accuracy.

These results are very preliminary:

- First of all, they are still mainly qualitative.
- Second, they only cover reusability, while OO methodology has other advantages, such as inheritance from class to class etc. (and, of course, this additional advantages may also take a toll on implementation complexity and running time).

Formulation of the Problem

For every possible input x, let us denote the desired output by $f(x)$. We have two alternatives:

- We can design a program that computes $f(x)$ for every x (this alternative corresponds to OO).
- Alternatively, we can design a program that computes $f(x)$ only for those x that satisfy a certain pre-condition $P(x)$. This pre-condition is usually computable (i.e., algorithmically checkable).

In order to make a reasonable choice, we must check whether with pre-condition, we can really get a simpler program.

What is Known About This Problem

This problem has been actively studied in the theory of computing, where problems with a fixed pre-condition are called *promise problems* (see, e.g., [4]); this name comes from the fact that we *promise* that when the algorithm will be used, all inputs x will satisfy the pre-condition $P(x)$.

The general question of comparing promise problems with the corresponding unrestricted problem is very difficult, and only for a few pre-conditions, we know the answer to this question.

Intuition of Software Developers: Why Use It and an Example

Since there are no *results* that are general enough for our task, we have to use, instead, the software developers' intuition, experience, and common sense.

One important part of this common sense knowledge (confirmed by experience) is as follows: whether we should use the pre-conditions or not depends on how complicated these pre-conditions are:

- If the pre-conditions are easy to describe and understand (e.g., they say that some values are positive, or that some other values belong to a certain interval, etc.), then such pre-conditions often help, and although there is no guarantee that using these pre-conditions will indeed lead to a simpler program, it is worth trying to simplify the program by imposing these pre-conditions.
- On the other hand, if the user describes his pre-conditions in highly technical and difficult-to-understand form, e.g., by saying that a certain integro-differential inequality must always be satisfied by the input data, then, for reasonably simple programs, it is easier to forget about these pre-conditions and write the most general possible module instead.

We will show that on the qualitative level, this intuition be formally justified.

Intuition Justified

Let $p_1(x)$ be a program that computes $f(x)$ for all x, and let $p_2(x)$ be a program that computes $f(x)$ only for those x for which $P(x)$ is true. It is quite possible that $p_2(x) = f(x)$ also for some x that do not satisfy the original pre-condition. In this case, we can say that the program $p_2(x)$ works correctly under a *weaker* pre-condition $P_w(x)$, namely, under a pre-condition that $p_2(x) = p_1(x)$.

To check this new pre-condition, it is sufficient to apply both programs $p_1(x)$ and $p_2(x)$ and check whether $p_1(x) = p_2(x)$. If both programs $p_1(x)$ and $p_2(x)$ belong to a certain asymptotic-time complexity class (see, e.g., [5]), e.g., if both require linear time, or both require quadratic time, etc., then the new pre-condition $P_w(x)$ also requires a similar computation time. Thus:

- if we have a program $p_1(x)$ that computes $f(x)$ for all x, and
- if the pre-condition $P(x)$ is more complicated than the function $f(x)$ in the sense that it takes asymptotically longer to check $P(x)$ than to compute $f(x)$,
- then our only hope for designing a faster (simpler, etc.) program $p_2(x)$ that computes $f(x)$ under the pre-condition $P(x)$ is to find a *weaker* pre-condition $P_w(x)$ that is easier to check, and to only use this easier pre-condition.

This conclusion is in perfect accordance with the above intuition: if no such weaker and easy-to-check pre-condition can be found, then we cannot simplify the program by using the pre-condition and therefore, we have to design this program to be as general as possible.

Often, Computation Time is Less of a Problem, but Accuracy is a Problem

In real-time system, the main objective is to be able to complete the computations *on time*: e.g., to compute the desired trajectory control before it is too late to use it. However, in many other problems, where computations are *off-line*, it is OK to spend an extra time on computations if this extra time will lead to better, e.g., more accurate computation results.

Case Study

A typical example of such a problem is the problem of extracting environmental and geophysical data from satellite images.

Towards the Mathematical Formulation of a Problem: Preliminaries

Let us denote by n, the total number of input values, and by x_i, the quantity measured in i-th measurement. Let y denote the quantity whose value we want to estimate as a result of these measurement, and let $\tilde{y} = f(\tilde{x}_1, \ldots, \tilde{x}_n)$ denote an algorithm that transforms the measured values $\tilde{x}_i$ of the quantities x_i into an estimate $\tilde{y}$ for the desired quantity y.

For example, for the environmental analysis of a satellite photo, n is the total number of pixels, x_i is the brightness of i-th pixel, and y is, e.g., the total pollution index.

When the signals x_i are strong, we usually do not have a problem of determining the desired values; in such situations, no serious filtering or data processing is needed. The real need for complicated data processing appears when we have *weak* signals $x_i \approx 0$. When $x_i \approx 0$, we can neglect terms that are quadratic, cubic, etc., in x_i and only consider linear terms. As a result, the general data processing algorithm $f(\tilde{x}_1, \ldots, \tilde{x}_n)$ can be written as a linear function: $\tilde{y} = f_0 + f_1 \cdot \tilde{x}_1 + \ldots + f_n \cdot \tilde{x}_n$.

Measurements are not 100% accurate; as a result, the measured values $\tilde{x}_i$ may differ from the actual values x_i. Due to these measurement errors $\Delta x_i = \tilde{x}_i - x_i$, the resulting estimate $\tilde{y}$ differs from the actual value y by the error $\Delta y = f_1 \cdot \Delta x_1 + \ldots + f_n \cdot \Delta x_n$.

The question is: can we decrease this error by using pre-conditions?

Towards the Mathematical Formulation of a Problem: Pre-conditions

In general, pre-conditions can be highly non-linear. However, since x_i are small, we can always assume that the pre-condition has the linear form $p_1 \cdot x_1 + \ldots + p_n \cdot x_n = p_0$.

Towards the Mathematical Formulation of a Problem: Statistical Accuracy

In this section, we will assume that the measurement errors are of statistical nature, and that we know the probabilities of different measurement errors. Usually, each measurement error is the result of a simultaneous action of several small factors, and therefore, by using the central limit theorem (see, e.g., [6]), we can conclude that each measurement error is normally distributed.

It is reasonable to assume that for each measurement, the average error is zero: otherwise, we can re-calibrate this measuring instrument and make it 0. So, we have a Gaussian distribution with 0 average for each measurement error Δx_i. It is known that such distribution is uniquely determined by the corresponding standard deviation σ_i.

Without losing mathematical generality, we can assume that all n measurements have the same accuracy, i.e., that $\sigma_1 = \ldots = \sigma_n = \sigma$; indeed, otherwise, we could consider new variables $x_i' = x_i \cdot (\sigma/\sigma_i)$ instead of x_i, and correspondingly re-define the coefficients f_i.

It is also reasonable to assume that errors of different measurements are independent random variables. In this case, the mean square deviation of Δy is equal to $(f_1^2 + \ldots + f_n^2) \cdot \sigma^2$.

If we take the pre-condition $\sum p_i \cdot x_i = p_0$ into consideration, then instead of computing y as $\sum f_i \cdot x_i$, we can alternatively pick a real number α and compute y as $\sum (f_i + \alpha \cdot p_i) \cdot x_i - \alpha \cdot p_0$. For this new algorithm, the resulting accuracy is equal to $\sigma^2 \cdot \sum (f_i + \alpha \cdot p_i)^2$.

- If this expression attains its minimum for $\alpha = 0$ or $\alpha \approx 0$, then using the pre-condition does not decrease the error. In this case, OO is better.

- If, on the other hand, the minimum is attained when $\alpha \neq 0$, then non-OO restriction does lead to a substantial error decrease. In this case, not OO is better.

When is OO Better?

The minimum of the above expression is attained for $\alpha = 0$ if and only if the derivative of this expression with respect to 0 is equal to 0, i.e., if $\sum f_i \cdot p_i = 0$. So, if we know the exact values of f_i and p_i, then whether we should use OO or not depends on whether this equality is true.

However, in practice, we often learn the exact values of f_i and p_i only at a certain advanced stage of software development. For such situations, it is desirable to have a general recommendation that would depend only on the number of measurements n.

In vector terms, the expression $\sum f_i \cdot p_i$ describes a dot (scalar) product of two vectors $\vec{f} = (f_1, \ldots, f_n)$ and $\vec{p} = (p_1, \ldots, p_n)$, and the condition

$\vec{f} \cdot \vec{p} = 0$ means that the vectors $\vec{f}$ and $\vec{p}$ are *orthogonal* to each other. Since we are assuming that we only know n, and that we do not know the vectors $\vec{f}$ and $\vec{p}$, we can assume that the vectors $\vec{p}$ and $\vec{f}$ are *randomly distributed* in some reasonable sense. Then, the question is: are two random vectors orthogonal to each other?

To answer this question, we must describe what "random" means. Orthogonality does not depend on the length of the vectors, only on their directions, so, without losing generality, we can assume that both $\vec{f}$ and $\vec{p}$ are independent *unit* vectors. In this case, the scalar product is simply equal to the cosine of the angle between these vectors. A unit vector is uniquely described by the corresponding point on a unit sphere; it is natural to require that the probability be invariant relative to arbitrary rotations; this property uniquely determines the probability as proportional to the $(n-1)$-dimensional area of the surface of the sphere.

When $\vec{f}$ and $\vec{p}$ are both unit vectors, their scalar product $\vec{f} \cdot \vec{p}$ can take any value between -1 and 1. The average value of this scalar product is 0 (because, e.g., we can replace $\vec{f}$ by $-\vec{f}$, the probabilities will remain the same, but the average will change the sign). To compute the standard deviation of $\vec{f} \cdot \vec{p}$, we can take into consideration the fact that if we rotate both vectors, then the scalar product remains unchanged. We can thus rotate the first vector so that it goes into $\vec{e} = (1, 0, \ldots, 0)$; then the rotated second one remains a random vector. Hence, the scalar product of two random vectors is equal to the scalar product $\vec{e} \cdot \vec{f} = f_1$ for a random vector unit $\vec{f} = (f_1, \ldots, f_n)$. The mean square value of this scalar product is equal to the mathematical expectation $E[f_1^2]$, and from $f_1^2 + \ldots + f_n^2 = 1$, we conclude that $E[f_1^2] = 1/n$. Thus, the standard deviation of the scalar product is equal to $1/\sqrt{n}$. Using three sigma rule from statistics, we conclude that this scalar product cannot exceed $3/\sqrt{n}$. Thus:

- When $3/\sqrt{n} \ll 1$, i.e., when $n \gg 9$, random vectors are orthogonal and OO is better (this is the case with satellite images, where n is indeed huge).
- On the other hand, for small n, non-OO may be better (and it can indeed be better; see, e.g., robotic application in [1]).

In many practical cases, we do not know the probabilities of measurement errors; we only know the *upper bounds* Δ_i on the errors (see, e.g., [2, 3]). In this case, similarly to the statistical case, we can assume, without losing generality, that $\Delta_1 = \ldots = \Delta_n = \Delta$. Under this assumption, the largest possible error Δy is equal to $(\sum |f_i|) \cdot \Delta$.

If we take the pre-condition into consideration, then we get $(\sum |f_i + \alpha \cdot p_i|) \cdot \Delta$ instead. Here, OO is also better when this expression has a 0 derivative for $\alpha = 0$, i.e., when $\sum \text{sign}(f_i) \cdot p_i = 0$.

One can show that for random vectors $\vec{f}$ and $\vec{p}$ this expression has a similar standard deviation as in the statistical case, and therefore, OO is also better for large n ($\gg 9$).

ACKNOWLEDGMENTS

This work was supported in part by NASA under cooperative agreement NCCW-0089, by NSF grants No. DUE-9750858 and No. EEC-9322370, and by Future Aerospace Science and Technology Program (FAST) Center for Structural Integrity of Aerospace Systems, effort sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant number F49620-95-1-0518.

The authors are thankful to Misha Koshelev and to Luc Longpré for valuable discussions, to George Corliss for editorial help, and to the anonymous referees for valuable advices.

REFERENCES

[1] Fuentes, O.L.; *et al.*; *Telemanipulation: the virtual tool approach and its interval-based justification*, These Proceedings.

[2] Kearfott, R.B.; *Rigorous global search: continuous problems*, Kluwer, Dordrecht, 1996.

[3] Kearfott, R.B.; Kreinovich, V.(eds.); *Applications of Interval Computations*, Kluwer, Dordrecht, 1996.

[4] Longpré, L.; Selman, A.L.; *Hard promise problems and nonuniform complexity*, Theoretical Computer Science, 1993, Vol. 115, No. 2, pp. 277–290.

[5] Papadimitriou, C.H.; *Computational Complexity*, Addison Wesley, San Diego, 1994.

[6] Wadsworth, H.M. (editor); *Handbook of Statistical Methods for Engineers and Scientists*, McGraw-Hill Publishing Co., N.Y., 1990.