

Re: Liskov Substitution Principle and Abstract Factories

Source: <http://coding.derkeiler.com/Archive/General/comp.object/2005-01/0986.html>

From: Mark Nicholls (nicholls.mark_at_mtvne.com)

Date: 01/18/05

Date: Tue, 18 Jan 2005 10:56:41 -0000

"Dmitry A. Kazakov" <mailbox@dmitry-kazakov.de> wrote in message
news:qukbppdmnots.lw5s8wh9kdms\$.dlg@40tude.net...

> *On Mon, 17 Jan 2005 14:21:24 -0000, Mark Nicholls wrote:*

>

> > *"Dmitry A. Kazakov" <mailbox@dmitry-kazakov.de> wrote in message*

> > *news:1vjlggk7b44f\$.5mjeps9qwold.dlg@40tude.net...*

>

> > > *Why? It is OO to me. It is still about types. At least according the*

> > > *definition of OO given by Robert Martin (OO = dynamic polymorphism) is*

> > > *perfectly OO!*

> >

> > *OK, but that's not my definition...or OO, I need 'objects', now in a
sense*

> > *you have inserted another layer of indirection between an object, and
the*

> > *functions that may be construed as 'methods'.....that's fine....I don't*

> > *believe I have seen this anywhere else...i.e. not in Rumbaugh, Booch,
Meyer,*

> > *GoF....and others....its not a criticism.*

>

> *I just see no other way to deal with all types uniformly. Why Boolean*

> *should be no object?*

As I say it's not a criticism.....I agree....my mind is expanded because if
this.

>

> > > > *If you do not know (or cannot formally analyse) the implementation of
the*

> > > > *program, then the program is a random variable.*

> > >

> > > *Really? Fermata theorem has been proven, but 5 years ago, was Fermata*

> > > *theorem = true random then?*

> >

> > *yep*

comp.object: Re: Liskov Substitution Principle and Abstract Factories

>>
>> *quite so.*
>>
>> *is the set of primes that differ by two bounded?*
>>
>> *I cannot answer you except to say, given a set of assumptions we can say it*
>> *is probably 99.999% true....this has been done, by mathematicians, in number*
>> *theory books.*
>>
>> *i.e.*
>> *is the set of primes that differ by two bounded?*
>>
>> *answer....I don't know, but I'm 99.999% sure it isn't.*
>
> *This is a statement about you, not about prime numbers. That's the point.*
> *Today you might be 99.999% sure, next day 2%. And BTW, uncertainty you are*
> *talking about is not random. It is fuzzy. It would be random if being*
asked
> *you would answer either yes or no. That's not the case.*

OK, I take the point.

I still stand by the statement.....in fact the question may be unprovable but true!....thus we can only say we are 99.9999% sure....or more or less depending on the statistical evidence we have.

>
>>>> *Is the weather deterministic?*
>>>> *or just so complex as to not submit to complete analysis.*
>>>
>>> *It is not because of complexity. The analysis we could do is based on the*
>>> *laws (thermodynamics) which are of statistical nature.*
>>
>> *Thermodynamics (as far as I remember) is not statistical.*
>
> *It is. Temperature, pressure, density, all are statistically defined*
> *values.*

I seem to remember lots of differential equations.....

I would assume that temperature is some sort of absolutely defined sum of kinetic energy of particles....but because we cannot practically observe that we make assumptions.

>
>> *If I knew the exact position of every single particle in the world, could I*
>> *not, exactly predict the weather.....(OK, there's a lot of problems*

with

> > *this statement, but as a thought experiment).*

>

> *That's mechanics.*

so?

>

> > *But we don't, so we take a statistical approach i.e.*

> >

> > *given I don't know the exact position, I can make all sorts of dubious*

> > *assumptions that makes their positions/temp/humidity into a statistical*

> > *observation.*

>

> *Maybe the mechanical view is comprehensive. But that is no matter. When*

you

> *model weather you are working within a statistical framework, i.e. with*

> *thermodynamics. We do not consider each particle,*

because it is impractical to do so (maybe like knowing the exact behaviour of a program).

With twin primes.....we do not know yet (and possible cannot know)

With the weather, it is impractical to know...

in both cases we resort to statistics to give some sort of indication of what we do.

> *we deal with sets of*

> *them. On the contrary when we write programs we consider each statement,*

> *each declaration etc. We do not treat them as random occurrences.*

rather like the prime numbers

> *It is*

> *questionable is something reasonable as thermodynamics could be deduced f*

> *we would.*

>

> *1. Our programs are too small for that (comparing the number of states of*

a

> *program and 1 cubic meter of gas)*

>

> *2. They are very instable. Small local changes lead to great difference in*

> *behavior.*

Rather like a program.

I did say there were lots of problems.....the chaotic nature make small errors magnify, I accept, but even so I 'know' its not going to be 29 degrees C tomorrow in London.

>
> *So I do not think that statistical approach might work here.*

I believe it is a weak tool, but it may be the only practical way to deduce some sort of measure of quality (i.e. correctness).

>
> >>> *not until you evaluate the implementation*
> >>>
> >>> $P(s = 4)$
> >>>
> >>> *S member of domain { 1+1, 2+1, 3+1 }*
> >>>
> >>> *s is a member of S with even distribution. (assumption!).*
> >>
> >> *An ungrounded one! Why 2+1 is as likely as 3+1? And there is a simple test.*
> >> *Start the program again. A truly stochastic system is not predictable.*
> >
> > *stochastic.....implies there is no link between S_n and S_{n+1}it does not*
> > *mean it is not predictable, in some statistical manner.....e.g. stochastic*
> > *random walks.*
> >
> > *i.e.*
> >
> > $P(S_{n+1}=4 \mid S_n = i) = P(S_{n+1}=4) = 1/4$
> >
> > *Right, you don't have $S_n=x$ you have $Pr(S_n=x)$. It renders anything to probability. You will never be able to make a firm prediction.*

But again, if it is impractical (it could be impossible) to 'prove' a program, so that's all we're left with.

>
> > *i.e. it is stochastic (i.e. it's conditional probability doesn't*
> > *change.....given a fixed statistical model).*
> >
> > *but it still submits to probability.*
> >
> > *i.e.. you make a set of observations, and build a model.*
> >
> > *given the model, each subsequent observation is assumed to be independent of*
> > *any previous one.....given the model.....but you need to run the*
> > *experiments to create the model.....and you can statistically query the*
> > *model via a hypothesis test.....i.e. how many observations do I need to*
> > *make before I believe/disbelieve that the model is correct and there are*
no

comp.object: Re: Liskov Substitution Principle and Abstract Factories

> > *other factors influencing S_n .*

>

> *No. Actually it is: how many observations you need to make before the
> probability of an error will be lower than epsilon. There is no way out.*

not with this....no what?.....no way out of what?

>

> > *But a program is perfectly predictable.*

> >

> > *Once it is completely observed, before the observation it isn't.*

>

> *No, it is a property of a program to be predictable. You know that $y=f(x)$,
> i.e. that for any given x there is one y . If f were random you could not
> say so. You would have $Pr(y|x)$ instead.*

you can!

For it to be random, does not mean it is utterly unpredictable.

Toss a coin....it comes out head or tail....it is random, yet there is only
1 answer and that answer is specifically defined to be a member of a set of
2.

>

> > *You do not know the concrete result,
> > but you know that it will be the same as yesterday.*

> >

> > *for a specific set of input { In }, we know that the output will be the
same*

> > *tomorrow....assuming we haven't missed one (e.g. time).*

> >

> > *We can make assumptions (based on observation about the nature of
> > programming teams), a team that consistently produces working code will
> > generally produce better results than a team that doesn't.....that is a
> > statistical observation....not logical proof....but an indicator.*

>

> *Yes, psychotherapy...*

so.....? If it's good enough to make a decision

e.g.

will this rocket get to the moon?

I am 99.9% sure it is.

Is that good enough

yes/no?

comp.object: Re: Liskov Substitution Principle and Abstract Factories

>
> >> *When you have to catch*
> >> *a lion, you could place a cage in the desert and just wait until it*
comes,
> >> *opens the cage, goes in and then bars the door. The probability of that*
is
> >> *greater than zero. But if you have a buggy program you can wait*
forever, it
> >> *won't become correct.*
> >
> > *I don't see the point....so what?*
> >
> > *I can ascertain the probability of a lion entering the a specific cage,*
by
> > *doing lots of experiments and observing how long it takes on average for*
the
> > *lion to enter, and then when you say*
>
> > *"how long will it take the lion to enter on average".....(or how long*
> > *will it take before the program goes wrong).*
>
> *Lion's behavior is random. Program's one is not.*

and primes are not.....yet I can say I have evidence thats says.....XYZ.....

> *You have to have some team*
> *changing the program. This would make the thing random. The team, not the*
> *program. There is no magic in a program, it comes only with a process*
which
> *involves people. You can use statistics to judge people, not programs. So*
a
> *careful statement will be kind of:*
>
> *Pr (Team X will make error Y performing operation Z | Coffee machine works*
> *& Managers are on vacations & ...)*

Pr (Team X will make not make an error such that this space rocket wont get
to the moon | Coffee machine works
& Managers are on vacations & ...)

yes.....this may be good enough.

>
> *Bayes knows how to get from that:*
>
> *Pr (P has bug | It was developed by team X)*

so what's the problem?

I accept it's weak, but it may be all we have.

>
>> *No, it is the same as follows. Let I write a number 1..100 on a piece of paper, but I do not show you. You have to guess it, according to some model (actually any model) in your head. Does this make the number random? Nope.*
>> *It is a solid, concrete number. The processes which lead you to a decision what number it might be are random, not the number itself. Same with the program. You can guess about it. Your guess might be random or not. But the program itself is still deterministically right or not.*
>>
>> *after observation, yes....before observation you either know nothing, or can 'guess' on the basis of a model.*
>>
>> *For example..which are the best lottery numbers to pick?*
>>
>> *surely they are all the same?*
>>
>> *wrong.*
>>
>> *some numbers are picked more than others (by the people guessing), so pick some numbers that other people don't and if you win, your expected win is greater than picking common, ones....i.e. the assumpton that people pick numbers at random is wrong.*
>
> *[But numbers are still same. You just changed the goal function.]*

I haven't changed anything?

>
>> *We could do the same experiment with you and the 1 to 100 game, we would probably find you pick low numbers more often than big ones.....i.e. I suspect 1 will come up more often than 73.....the number is random (before the observation) and not completely evenly distributed.....programmers are probably similarly predictable, and so is their software.*
>
> *I do not object that one could consider results of a team of developers as a random variable.*

and that is the program.

comp.object: Re: Liskov Substitution Principle and Abstract Factories

- > *But the jump from random behavior of people to random*
- > *behavior of programs to too big too me. I do not believe it could be*
- > *feasible.*

I do not see the problem.

A team of mathematicians write formula to prove the twin prime conjecture.

$P(\text{team proves it}) = P(\text{the proof is correct})$.

before observation of the proof.

$P(\text{team writes some perfect code}) = P(\text{code is correct}) (=0)$

- >
- > *again we cannot equate logic with probability.*
- > >
- > $p(\text{program works} \mid \text{good programmer}) > p(\text{program works} \mid \text{bad programmer})$
- >
- > *The same program? In the statement above programs are two different*
- > *programs.*

from the set of programs....(they cannot be the same program, I am assuming bad/good are mutually exclusive....though they could *after inspection* be isomorphic).

- > *Then I do not believe that the confidence factor $p()$ above is*
- > *probability. People tend to make same errors over and over again. Should a*
- > *bad programmer implement the same problem 10 times the results will not*
- > *form nice Gauss distribution, we know it too well.*

Yet I still would choose to give hard programs to good programmers and easy ones to bad ones....because I know that the probability of getting a good enough result is best served that way.

- >
- > *If you are correct the above statement is wrong.....*
- >
- > *It isn't wrong, it is just unclear what it formally means. It would be*
- > *very*
- > *difficult, if possible to formalize it.*

would it.....

good mathematician = mathematician who passed BSc in maths with a first.
(dubious)

bad mathematician = mathematician who passed BSc in maths with a third.
(dubious)

$P(\text{mathematician can prove a theorem from number theory} \mid \text{good}) >$
 $P(\text{mathematician can prove a theorem from number theory} \mid \text{bad})$

comp.object: Re: Liskov Substitution Principle and Abstract Factories

We could easily take 100 mathematicians and give a statistical indication of how likely the above sentence is true....

Even if the mathematicians have produced the proof (before observation of actual proof)....

$P(\text{proof is correct} \mid \text{good}) > P(\text{proof is correct} \mid \text{bad})$

Is programming such a leap from this.

>
>>> *A true statistical model would be a random code generator. Then you have to*
>>> *decide whether the code is correct. We could agree that is far more easy*
>>> *just to throw that garbage away and write it from scratch! (:–))*
>>
>> *People are not random code generators though.*
>>
>> *Even if they are we could probably make some assertions.*
>>
>> *e.g.*
>>
>> *x member {1...10}*
>>
>> $S=x*x$
>>
>> *whats the expected value of S? or even x?*
>>
>> *to say that they are random (evenly distributed) is a highly powerful*
>> *statement, upon which we can make highly poweful predictions.*
>
> *If you know the outcomes {1..10}. As I said it is difficult to present a*
> *set of outcomes for a program P.*

Isn't this it's contract?

> *Even more difficult is to judge about its*
> *distribution.*

If the program is produced randomly yes, if it's produced by people no.

>
>>> *Same with the*
>>> *programs. You saw the design, you know the programmers, you designed*
the
>>> *tests etc. It is not statistics. For good, I'd say, otherwise nothing*
would
>>> *ever work!*
>>
>> *I don't follow the last bit.*

comp.object: Re: Liskov Substitution Principle and Abstract Factories

>
> *People do not write programs randomly. We have agreed on that.*

exactly, we can predict people quite effectively.

>
> >> *That (Cartesian product) does not warranty substitutability. C-E is the
> >> example. Derive Ellipse from Circle and you will see. [In other post I
am
> >> trying to explain why Circle->Ellipse mapping, or Real->Complex one are
not
> >> enough for substitutability. In general case you will need
isomorphism.]*
> >
> > *OK....not bijective isomorphism surely?*
>
> *Bijective!*

It's been a while since I used these terms

I'm confusing homomorphisms, isomorphisms and monomorphisms.....

OK for

A to be substitutable for B in all possible contexts then yes.

But surely not in a specific context.....though now I can't remember why
we're talking about it.

>
> > *i.e. I can substitute complex for real*
>
> *You cannot in $+: R \times R \rightarrow R$. But you can in $Read: File \rightarrow R$*

this is what's confusing to me.

// dispatching on all parameters.....? is that sensible...or even correct
terminology.

```
number Add(number x, number y)
{
    return x + y;
}
```

in all context for real numbers...i.e, where $+: R \times R \rightarrow R$ goes to $C \times C \rightarrow C$

```
void ClientAdd()
{
    Real r1 = 1;
    Real r2 = 2;
```

```
Real r3 = r1 + r2.  
}
```

I can substitute complex for real (actually this is similar to the OP question really).

>
> >.....*but not real for complex?*
>
> *You can in $+ : C \times C \rightarrow C$. But you cannot in $Read : File \rightarrow C$.*
>
> > *because there exists isomorphism from real to complex....but not vice versa?*
>
> *To make all possible substitutions work you need a bijection. So in the most general case you need $R \rightarrow C$ and $C \rightarrow R$. That is the problem.*

OK, let me just get the above example into my head, such that we can both agree and I will move on.

>
> --
> *Regards,*
> *Dmitry A. Kazakov*
> <http://www.dmitry-kazakov.de>