

Re: OOP/OOD Philosophy

Source: <http://coding.derkeiler.com/Archive/General/comp.object/2005-07/msg00150.html>

- *From:* "krasicki" <Krasicki@xxxxxxxx>
 - *Date:* 7 Jul 2005 08:32:57 -0700
-

Les Cargill wrote:

> krasicki wrote:

>>

>> Michael Feathers wrote:

>>>

>>> krasicki wrote:

>>>>

>>>> Daniel Parker wrote:

>>>>>

>>>>>

>>>>> "krasicki" <Krasicki@xxxxxxxx> wrote in message

>>>>> news:1120611204.476214.162240@xx

>>>>>>

>>>>>>

>>>>>>

>>>>>>> Look, OOD is about designing theoretical systems ...

>>>>>>>>

>>>>>>>> Just out of curiosity, what theory?

>>>>>>>>>

>>>>>>>>> — Daniel

>>>>>>>>>>

>>>>>>>>>>

>>>>>>>>>> Well Daniel,

>>>>>>>>>>>

>>>>>>>>>>> Not all systems consist of a web front end that accesses data from a

>>>>>>>>>>> well-established database though many do.

>>>>>>>>>>>>

>>>>>>>>>>>> OOD, depending on the tools and techniques you use, allows system

>>>>>>>>>>>> designers to model potential solutions in numerous ways. If this

>>>>>>>>>>>> tradeoff is made here, this benefit is forthcoming there and so on.

>>>>>>>>>>>> Design is about applying recombinant ideas to solving problems.

>>>>>>>>>>>>>

>>>>>>>>>>>>> Systems are developed to solve problems in effective ways. Systems are

>>>>>>>>>>>>> not developed for the sake of software or to gratify the provincial and

>>>>>>>>>>>>> esoteric ego needs of the employees charged with getting the system

>>>>>>>>>>>>> implemented.

>>>>>>>>>>>>>>

Re: OOP/OOD Philosophy

> >>>>I'll give you a good example. A Hilton was just built in Hartford and
> >>>>the building went up fast to meet the deadline of the new convention
> >>>>center being built next door. All federal guidelines were applied.
> >>>>
> >>>>So come inspection day, Connecticut's inspectors applied Connecticut
> >>>>building statutes to the inspection and the building failed! Now one
> >>>>could say that the building was built fast, saved money, and looks real
> >>>>good and boy were the builders proud of it. Let's call this process
> >>>>agile.
> >>>>
> >>>>So sixteen of the rooms were incorrectly built for handicapped access,
> >>>>an error of three inches per room. Where to get three inches? Push
> >>>>the rooms into the hall and the hall fails. You can see how this goes.
> >>>>
> >>>>Another example. A Hospital builds a new wing onto an existing
> >>>>structure and these days new hospital wings look like fancy hotels.
> >>>>Everything is immaculate, grand fascades, fancy everything. The
> >>>>hospital wing opens without a hitch.
> >>>>
> >>>>The first bed is rolled down the hall, the elevator button is pushed,
> >>>>the door opens, the bed pushed into the elevetor as far as it can go,
> >>>>but the bed still doesn't fit. Wrong sized elevator.
> >>>>
> >>>>
> >>>>It's pretty amazing to me that you find anything in common with Agile in
> >>>>these scenarios. They all sound like cases were there was no feedback
> >>>>or testing. Sounds more like plan-driven development to me.
> >>>>
> >>>>
> > Au contraire. The bricks all passed unit tests. As did the cement,
> > steel, and so on. And the plans all had feedback.
>
> Not the right feedback. It's a failure of requirements capture,
> pure and simple. No methodology nor any other thing, other
> than collecting all the relevant requirements and checklisting
> them, would have made a bit of difference.

I will infer that you're saying that More planning and preparation
time might have comprehensively accumulated and accounted for these
missing design considerations. In other words a heavier weight
methodology could have avoided the headaches – all things being equal.

>
> > And the customer
> > surely showed up with a glowing smile watching the obvious progress.
> > And progress happened every day.
> >>
> > The elevator worked fine. Up. Down. Ring, ring. All positive
> > feedback.
> >>
> > The workers sweated. The execs wore suits and went golfing.

Re: OOP/OOD Philosophy

>>
>>
>>>> These are true stories. Shouldn't all of the architects of these
>>>> buildings have expected change to happen as well. Same with the
>>>> builders. maybe build with everything loose so that it can be
>>>> reassembled when the next minor detail arises? Aren't we being told
>>>> this is the way things work?
>>>
>>>Well, the fact is software is malleable. In fact it is too malleable.
>>>It isn't hard to change software at all. All you have to do is type a
>>>couple of characters in any program and you can break it. Because that
>>>is the way that software is, we need tests to give it backbone.
>>
>>
>> It's not that malleable. Once in production software is very hard to
>> change for all kinds of political reasons.
>>
>
> It shouldn't go into production with defects that are gonna
> cost people money, at least without an enforceable plan
> to get the defects out, upfront.
>
> Once in production, somebody has to make the decisions of
> when, how and why to deploy upgrades.

Well, my point is that if something goes through the XP methodology with all of the hot air and hubris that one has performed a bazillion tests on it already but defects still exist, who will know and how could they prove it.

Would any of us argue for long with these people? I lose heart just trying to get a straight answer out of them in something as straightforward as a newsgroup. Haven't you heard, XP is absolutely right because they've tested everything every which way.

Once in commercial production, software that is mission critical is not easily changed because, as someone said elsewhere, tweaking the wrong bit could cause system calamities. Reintroducing code in these environments could take six months to a year of expensive rework or total shutdown. It's no longer a question of tweaking code but questioning all assumptions. With BDUF, you can isolate the problem and hypothetically run the system without software trying to understand the overall implications.

>
>> In fact a big problem for architects and designers is having
>> programmers undermine design activity with too much dog and pony
>> prototyping.
>
> How is that possible? Other than time being wasted, prototyping
> is harmless. Prototypes should not even be attempted until

Re: OOP/OOD Philosophy

- > there's a specific question or suite of questions they are
- > to answer. If it's a sandboxed protptype, just to let the
- > programmers play, then chunk it, or put it away. You
- > still need specific deliverables from the prototyping.

Prototyping is political dynamite in many organizations. Software designers and architects are usually discussing issues that are not near and dear to the hearts of the local application domain princess who wants to have someone to talk to. Enter, any number of local characters who begin prototyping their idea of what should happen. Before long the architects and designers are entangled in favorite color discussions and presentation fashion shows.

Add to this mix, any number of programmers who believe they know better than the people always talking about abstract ideas and you enter the realm of random, esoteric, and uncontrollable development.

- >
- >> Bad ideas become adopted before any discussion of the
- >> larger picture can be formulated.
- >>
- >
- > Then they have to get rooted out and killed, or at least
- > triaged and weighed for "badness". Bad ideas that don't get
- > shot are a sign of complacency, not methodology.

There is no budget to root things out and bad software is often sponsored internally by incompetent people who control your paycheck. XP adds authenticity to the problems involved.

Because software development is so tightly coupled to the individuals, it is no longer a matter of correcting or eliminating problematic code. The XP crowd has a vociferous ego stake in what's being done. They've got stories and tests and feedback loops that will insist it's there right. They all feel good about it. And there is no impartial design document you can point to to say otherwise because the whole ball of wax is personal, intimate, immediate, and a treadmill of exhaustion for everyone involved.

- >
- >> Of course you need tests. We aren't a bunch of ninnies here.
- >>
- >>
- >>> Even carpenters measure before they cut. Yet, in computer science we
- >>> are being told that we should operate as though we are all alcoholics
- >>> and take things one day at a time.
- >>>
- >>>The problem is: misunderstanding the material you are working with.
- >>>Code is not wood or concrete.
- >>
- >>

Re: OOP/OOD Philosophy

- >> But spent resources are. Nobody fixes anything for free. And bad code
- >> applied to millions of daily transactions can cost companies or
- >> customers lots and lots of money when wrong.
- >>
- >
- > So somebody has to do a cost-benefit analysis of when to do what.
- > Good code isn't free, either. This is logistics, not particularly
- > even software logistics.

The key term is "has to".

- >
- >> Testing is tricky stuff and complex logic errors don't get discussed
- >> when daily iterations are the norm because there is no time.
- >>
- >> Design and OOD are not code or code design.
- >>
- >
- > You can't fix culture with tools, in other words. Mostly, yes :)
- >

Thanks Les. Arguing XP is as thankless a task as I've ever encountered. The proponents swarm on critics like hornets so try to avoid this stuff more often than not. I sincerely was trying to give the OP a fair assessment of what's out there but this quagmire blocks all light from shining through.

.

• *Follow-Ups:*

- ◆ **Re: OOP/OOD Philosophy**
◇ *From:* Robert C . Martin
- ◆ **Re: OOP/OOD Philosophy**
◇ *From:* Les Cargill

• *References:*

- ◆ **Re: OOP/OOD Philosophy**
◇ *From:* krasicki
- ◆ **Re: OOP/OOD Philosophy**
◇ *From:* Michael Feathers
- ◆ **Re: OOP/OOD Philosophy**
◇ *From:* krasicki
- ◆ **Re: OOP/OOD Philosophy**
◇ *From:* Nick Malik [Microsoft]
- ◆ **Re: OOP/OOD Philosophy**
◇ *From:* krasicki
- ◆ **Re: OOP/OOD Philosophy**
◇ *From:* Nick Malik [Microsoft]
- ◆ **Re: OOP/OOD Philosophy**
◇ *From:* Robert C . Martin

- ◆ **Re: OOP/OOD Philosophy**
◇ From: krasicki
- ◆ **Re: OOP/OOD Philosophy**
◇ From: Daniel Parker
- ◆ **Re: OOP/OOD Philosophy**
◇ From: krasicki
- ◆ **Re: OOP/OOD Philosophy**
◇ From: Michael Feathers
- ◆ **Re: OOP/OOD Philosophy**
◇ From: krasicki
- ◆ **Re: OOP/OOD Philosophy**
◇ From: Les Cargill

- Prev by Date: **Re: Question regarding OOP and database access**
- Next by Date: **State Machine/Class Granularity**
- Previous by thread: **Re: OOP/OOD Philosophy**
- Next by thread: **Re: OOP/OOD Philosophy**
- Index(es):
 - ◆ **Date**
 - ◆ **Thread**