

Re: Relational-to-OOP Tax

Source: <http://coding.derkeiler.com/Archive/General/comp.object/2007-02/msg00314.html>

- *From:* Thomas Gagne <tgagne@xxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Mon, 19 Feb 2007 13:57:34 -0500
-

frebe wrote:

Decoupling SQL statements may or may not increase the total number of lines of code, depending on the size of the SQL statements and the number of times they are reused.

Forcing all SQL statements into functions instead of only putting them into functions when suitable, will always increase LOC.

As discussed elsewhere in the thread, even the hard of learning Mr. Jacobs admits that lines of code is not a sufficient measure.

More lines of code takes longer time to write. More lines of code makes the code harder to read.

This is not always, or even often, the case. Highly coupled big balls of mud that mix different concerns are more difficult to understand than well-factored, clean code.

But nobody have proved why separating all SQL statements would create cleaner or more well-factored code.

You keep asking for proof without specifying the burden of proof. What burden of proof would require for structured programming to prove it is better to call functions than not to (as used to be the case) or that it is better to let a filesystem handle file-io than an application to manipulate disk structures directly (as also used to be the case)? Is the burden of proof really so different that you can't extrapolate your experiences calling functions or using open/read/write/close to manipulate files onto SQL?

<snip>
Maintainability and extensibility are two NFRs that come immediately to mind. Clean code is easier to maintain and extend.

Re: Relational-to-OOP Tax

In what way is

```
createEmployee(x,y,z)
```

```
function createEmployee(x,y,z) {  
  insert into employee values (x,y,z)  
}
```

cleaner than

```
insert into employee values (x,y,z)
```

It is cleaner when createEmployee(x,y,z) is more realistically imagined:

```
insert into employee...  
insert into address ...  
insert into employeeAddress ...  
insert into employeeEmployee ...  
insert into calendarResource ...  
insert into payroll ...  
insert into planParticipant ...  
insert into benefitParticipant ...  
insert into dependents ...
```

and when another table is created that must know about new employees the procedure can be updated without modifying application code. Or if employees can be added from multiple applications it is better to have isolated the code for adding them than to duplicate it.

<snip>

Are you arguing that clean, well-designed code does not have quality benefits?

No, but creating new functions with only one statement or functions that are only called only once, doesn't create cleaner or more well-designed code.

Practically speaking that is rarely the case. And even if it occurred more often than rarely, it is a better design that allows one module/function/method to change independently of others that rely on it.

—
Visit <http://blogs.instreamfinancial.com/anything.php> to read my rants on technology and the finance industry.

.