4 Oct 2017

OO programming

In object-oriented (OO) programming, a class is an extensible program-code-template for creating objects, providing initial values for state (member variables) and implementations of behaviour (member functions or methods). The following is an example of an object called a 'Button' which, when placed on a screen (another object class) may look like something like this

In the OO world this button contains the following properties:

Button
- xsize - ysize - label_text - interested_listeners - xposition - yposition
+ draw() + press() + register_callback() + unregister_callback()

This button is an artifact that is placed on a screen and when pressed by an operator, it will cause some other object (also known as a code object) to be executed. You will notice the property called 'press()'. Whenever this particular button is pressed, the code object associated with this property will be executed.

The question that now needs to be asked is: when is an operator likely to be called upon to press this specific button?

More importantly is it the programmer's responsibility to decide as to when it is to be pressed or is there a specific business requirement calling for this button to be pressed?

In any computerised business system there are literally thousands of such objects that have to be accounted for and certainly not by a programmer of code.

Allow me to give you another example. Imagine a banking application where 5 such buttons are placed on a screen enabling an operator to choose the type of banking account that the person wants to open. Each button will be labelled with an appropriate name, for example: Savings account; Cheque account; Debit account; Credit account; Cash management account.

It is now very simple for the programmer to have a single 'press()' function for every button. All that is required in the press() property is for the button to call a routine passing a parameter of the appropriate button. So, for example, the function could be press('OpenAccount',1) for the first button (which will open a new savings account), press('OpenAccount',2) for the second button (which will open a new cheque account) etc.

Seems very simple does it not.

Now ask yourself the question, who decided that the business needed the 5 different banking accounts? Certainly not the programmer nor the database designer.

So where in the enterprise architecture lifecycle is this decision made? Is it:

1) An ‘objective’? Perhaps, but what sort of objective? A value-chain? A customer requirement?
2) A ‘strategy’? Perhaps, but what sort of strategy? Open a banking account?
3) A type of ‘data’? Well in OO every object is given the property of ‘data’ and somehow every enterprise architecture approach seems to leave this major decision up to the IT department

Without a clear understanding of the business ‘knowledge’, the final arbitrator of the IT system will lie in the hands of non-business operatives, namely DevOps.

The following is an example of what business knowledge would be needed to solve this enigma:

1) Every account type is a sub class of 'account' which is a type of ‘service’ which is a type of ‘offering’
2) The ‘open-account-document’ is a type of ‘account-handling-document’ which is a type of ‘non-financial-document’ which is a type of ‘document’
3) A relationship link is established via another knowledge class called the ‘non-financial-document-line’ which will link the non-financial-document to it and the ‘service’
4) These classes of knowledge are linked to the performance indicator called ‘Number of accounts’ (a class of business objective called a ‘measure’) which supports the ‘value’ (a class of business objective called a ‘goal’) of ‘profitability’ which supports the businesses ‘benefit’ (another class of business objective called a ‘goal’) of ‘prosperity’ which supports the business’ ‘purpose’

Can anyone allow this BITG (business IT gap) to continue unabated by the implicit and imprecise use of enterprise architecture supported by the non-business expertise of IT professionals?

Everyone in the enterprise has a role to play but without an explicit business ‘information’ model these roles will forever be blurred.

Regards

Charles Meyer Richter
Principal information architect and Diagnostician
Ripose Pty Limited
charles.richter@ripose.com

ps you may have noticed I have not addressed the ‘customer’ knowledge class as this is another ‘can of worms’ just about every database designer has trouble designing. Not surprising though as most business operatives may find it hard to define the ‘customer’.