Design By Contract (DbC)

Article
Herman Slagman · Jul 21 5m read

Design By Contract (DbC)

This time I don't want to write about a brilliant feature of IRIS (of which it has many), but instead about a feature that is sorely missing.
Today, talking about OOP is not sexy. Although almost all modern programming languages implement some kind of OOP, discussions about fundamental issues of software development are not very common between real-world implementors of such technologies as developers are. In fact, Computer Science as a whole isn't a mainstream topic between developers, which I think it should be.
In this InterSystems Developers Community, most posts are about questions (How do I … ?) of how to use the technology or practical solutions to a problem (How can I … ?).
But not many are about fundamental computer science concepts and how they are implemented (or not) in our favorite Data Platform.

How can one be using a fundamental concept such as OOP, but not discussing it how it is being implemented in your favorite platform?
It is the same with everybody using (and having opinions about) REST API's without even knowing what REST means or have read Roy Fielding's dissertation.
Or using the concept of an ESB and know nothing about Dave Chappel's book where he first introduced the term Enterprise Service Bus.
(Mis)Using the term 'agile' without ever having read the Agile Manifesto.

One of my all-time favorite books about computer science and OOP is Bertrand Meyer's book Object-Oriented Software Construction (2nd Ed) (OOSC), a classic about software quality, robustness, extensibility, and reusaibility, and why OOP is a solution for many of these issues.
Every developer who is using classes, objects, methods, and inheritance should read this book.

Based on the work of DataTree, InterSystems has made an excellent OOP implementation with Cache and the Cache Object Script extensions to Mumps. Great multiple inheritance (of which I will probably write a different article) and polymorphism, not only in the programming language but also in the database. In my opinion, the only (but unique) combination that truly implements OOP: code and data in one place.

But what IRIS lacks and which is described in detail in OOSC, is a feature that lays a concise robustness layer upon OOP which is called Design by Contract.

What we often see in code (methods) is:
method getMail(id) ? Mail
  If not id return "Id is required"

This is what we call 'defensive programming', the method being called is arming itself against 'illegal' invocations. But the 'contract' of the method is obvious: if you give me an (valid) id I will return the associated email message, this code, executed at every invocation, shouldn't be necessary. This way the responsibility is being shifted from the caller to the method being called.

DbC is all about that, the 'contract' of a class or a method or an object instance of that class states precisely what is expected as input and what is promised as output as long as the caller adheres to the contract of the method (or the class as a whole).

DbC is defensive programming on steroids, without the runtime penalty of checking everything: you describe all constraints at design time.
DbC has two runtime modes: development and production.

In development mode, all constraints and checks are executed and reported.
In production mode, they will only be executed in case of an exception.

How does DbC do that?

At a class level, you define ‘invariants’: conditions that need to be met at all times (entering constructor methods excluded).
At a method level, you define pre-conditions and post-conditions.
Before even executing a single line of code, the pre-conditions must be met.
After having executed the method’s code the post-conditions must be met, for that for every property there is an ‘old’ value, the value the property had before the method was executed.
And for both entering as well as exiting the method, the invariants must still be valid.

In development mode, these conditions will all be executed. At that level you can choose how to handle unfulfilled conditions: throw an error or just report the event.

In production mode, the conditions will not be executed, but in the advent of an exception within the method, both invariants and pre-conditions will be checked and are part of the exception info, after that an error is being raised at all times.
You can also define a ‘rescue’ method, code that is being executed when an exception occurs, from setting database values, closing files and connections, executing repair code, etc. You can even set a ‘try-again’ flag.
Of course, the latter you can also achieve with IRIS standard $ZTrap or Try-Catch mechanisms.

Why should we want DbC in IRIS Objectscript?
I think the code will be much more robust and, having not to worry about performance penalties at runtime, the conditions can (should) be quite elaborate.
Having constructed the right set of conditions, unit-tests would hardly be needed, the ‘tests’ will be run at every invocation of a method. And, in contrary to separate unit-test frameworks, the conditions are right in the code, it’s documentation too.

If DbC is so great why isn't it widely applied?
Coming back to my introduction: ignorance.
As developers, in general, don’t ‘do’ Computer Science, they will never be able to grasp the essence of fundamental computer engineering concepts. But, even worse, educators, professional training programs, and ‘experts’ are hardly aware of the DbC concept.
(Unit) Testing is a far more easy concept to understand than writing a ‘contract’ for classes or code.
I also expect it to be quite difficult to build into an existing language or compiler.
There are a few languages that have DbC built-in, most prominent is Eiffel, a language, not entirely surprising, designed by Bertrand Meyer.

At InterSystems the trend seems to be slowly moving away from ObjectScript, especially with the upcoming Embedded Python feature, which is a brilliant move.
But when ObjectScript could have DbC built-in, the language would get a tremendous boost for designing robust applications.
It would be interesting to learn what the ObjectScript core developers have to say about this matter.

#InterSystems IRIS
70 2 1 1 151