"LET IT CRASH" MEETS "IT SHOULDN'T CRASH"

DESIGN BY CONTRACT IN ELIXIR

@GUILLEIGUARAN

GUILLERMO
IGUARÁN SUAREZ
RIDE.COM

@ELBASANCHEZM

ELBA SÁNCHEZ
MÁRQUEZ
RIDE.COM

ride

#ONELESSCAR

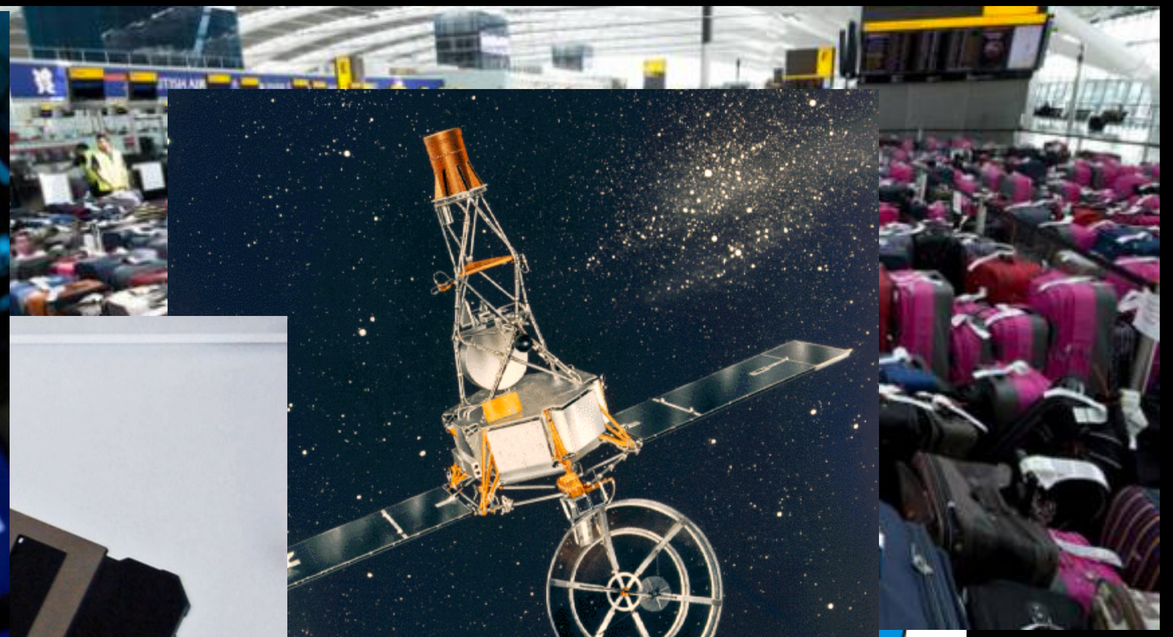
HOW DID WE GET HERE

BUGS AND CRASHES

DID YOU KNOW?


Knight
NYSE
Knight
Museum

Internet Worm -
Source code
X1294.96A-D




2 km
5 km
Kourou
15 km

ARIANE 5 FLIGHT 501

- THE ARIANE 5'S FASTER ENGINES EXPLOITED A BUG THAT WAS NOT FOUND IN PREVIOUS MODELS.

- THE SOFTWARE HAD TRIED TO CRAM A 64-BIT NUMBER INTO A 16-BIT SPACE.

- THERE WAS NO EXPLICIT EXCEPTION HANDLER TO CATCH THE EXCEPTION, SO IT FOLLOWED THE USUAL FATE OF UNCAUGHT EXCEPTIONS AND CRASHED THE ENTIRE SOFTWARE, HENCE THE ONBOARD COMPUTERS, HENCE THE MISSION

YOU CAN'T

BLAME MANAGEMENT

YOU CAN'T

BLAME THE LANGUAGE

YOU CAN'T

BLAME THE IMPLEMENTATION

YOU CAN'T

BLAME TESTING

YOU HAVE TO BLAME

THE REUSE SPECIFICATION

The requirement that the horizontal
bias should fit on 16 bits was in fact
stated in an obscure part of a mission document.
But it was nowhere to be found in
the code itself!

A LITTLE BIT OF THEORY

HISTORY

- The pre- and postcondition technique originated with the work of Tony Hoare, whose 1969 Communications of the ACM paper described program semantics using such assertions.
- The term was coined by Bertrand Meyer in connection with his design of the Eiffel programming language

Design by contract has its roots in work on formal verification, formal specification and Hoare logic.

BASICS OF HOARE LOGIC

- Formal reasoning about program correctness using pre- and postconditions

Hoare Logic is at the core of the deductive approach of the DbC.

“Design by Contract“ falls under Implementation/Design

- Typically unit tests are used to verify that the software works correctly under certain example cases but hardly can be used to detect all possible edge cases during development.

- Design by contract (DbC) is a software correctness methodology that document and programmatically assert the change in state caused by a piece of a program

IN ARIANE'S CASE

Where the precondition (require...) states clearly and precisely what the input must satisfy to be acceptable.

```
convert (horizontal_bias:
DOUBLE): INTEGER  is
require
   horizontal_bias
      <= Maximum_bias
do
   ...
ensure
   ...
end
```

Eiffel Version

WHAT IS DESIGN BY CONTRACT?

- Each party benefits and accepts obligations
- One party's benefits are the other party's obligation
- It is described so that both parties understand what would be guaranteed without saying how.

	OBLIGATIONS	RIGHTS
PASSENGER	BUY AIRLINE TICKET, BRING ACCEPTED BAGGAGE AND BE AT AIRPORT 2 HOURS BEFORE	REACH DESTINATION
AIRLINE	BRING PASSENGER TO DESTINATION	- NO NEED TO CARRY PASSENGER WHO IS LATE - OR HAS UNACCEPTABLE BAGGAGE - OR HASN'T PAID TICKET

STRUCTURE OF A CONTRACT

PRECONDITION (**REQUIRES CLAUSE**)

POSTCONDITION (**ENSURES CLAUSE**)

- If its precondition is true when a method is called, then the method will terminate return to the calling program — and the postcondition will be true when it does return.
- If its precondition is not true when a method is called, then the method may do nothing

```
put (x: ELEMENT; key: STRING) is
                -- Insert x so that it will be retrievable through key.
        require
                count <= capacity
                not key.empty
        do
                ... Some insertion algorithm ...
        ensure
                has (x)
                item (key) = x
                count = old count + 1
        end
```

https://www.eiffel.com/values/design-by-contract/introduction/

HOW CAN IT HELP?

“When quality is pursued, productivity follows.”

–K. FUJINO
VICE PRESIDENT OF NEC CORPORATION’S C&C SOFTWARE DEVELOPMENT GROUP

AND ALSO WE LOOK FOR...

RELIABILITY

Correctness

Robustness

- Assertions (preconditions and postconditions
in particular) can be automatically turned on during testing

- Assertions can remain turned on during
execution, triggering an exception if violated

- Most important, assertions are a prime component of the software and its automatically produced documentation

LANGUAGE SUPPORT

- There are several implementations of DbC libraries for some languages

- And languages with native su

AND NOW WITH ELIXIR

METAPROGRAMMING IN ELIXIR

Book by Chris McCord - O'Reilly

MACROS RULES

- Rule 1: Don’t write Macros

Book by Chris McCord - O'Reilly

MACROS RULES

- Rule 2: Use Macros Gratuitously

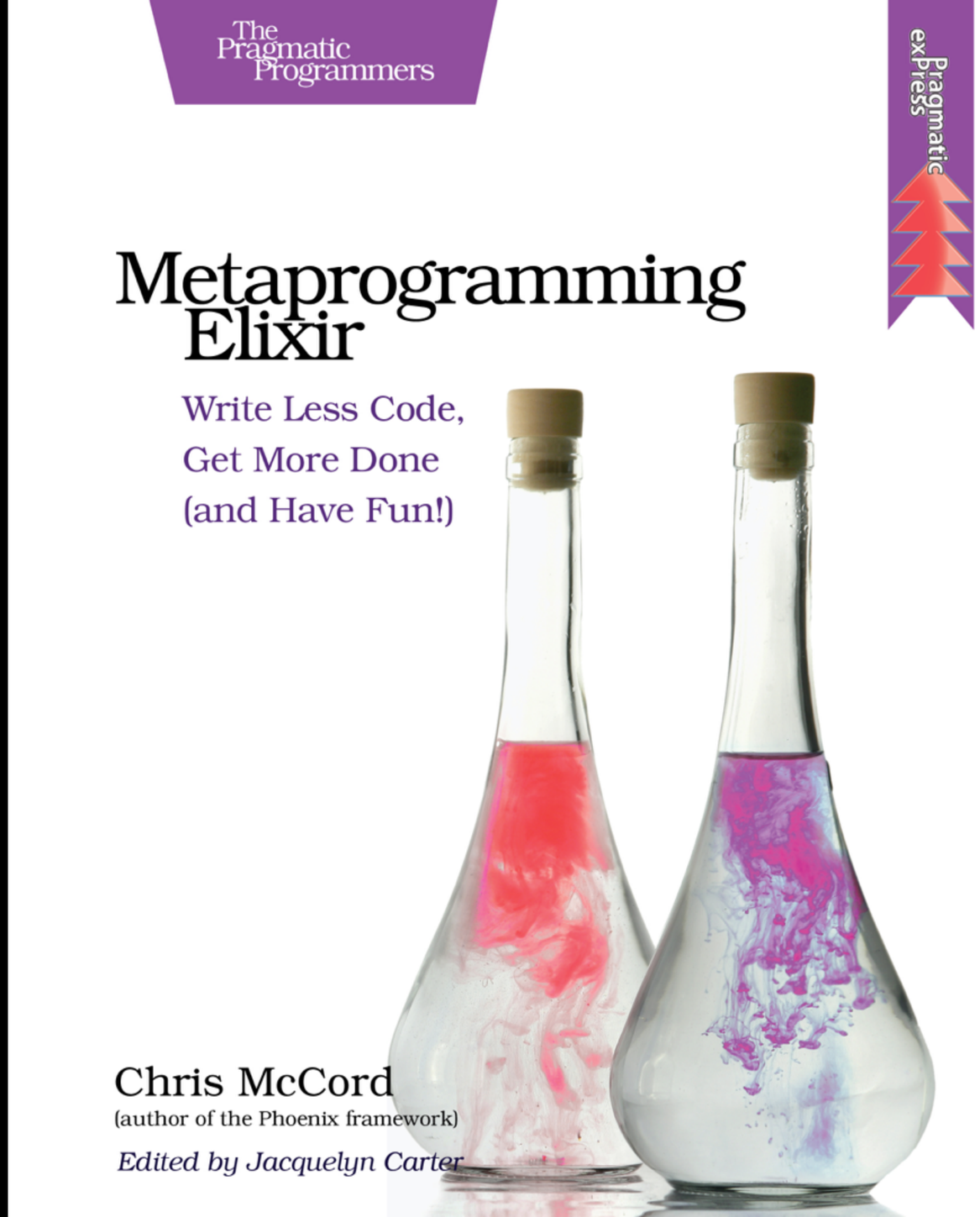
Book by Chris McCord - O'Reilly

MACROS

- A macro is code that writes code
- Many constructs in Elixir are macros (def, if, unless, defmodule,…)
- Elixir code runs at compile time and can be used to manipulate language AST.

METAPROGRAMMING ELIXIR BY CHRIS MCCORD

Book by Chris McCord - O'Reilly



BACK TO DBC...

- We will use Elixir macros to extend the language adding support for basic DbC constructs.
- We will tag existing functions with “requires” and “ensures” tags.
- Macros will manipulate function body to insert precondition and postconditions inside of functions.

WHAT WE HAD TO DO

```elixir
defmodule Math do
  use Contracts

  requires num >= 0
  ensures result >= 0 && :math.pow(result, 2) <= num && :math.pow(result + 1, 2) >= num
  def sqrt(num) do
    result = :math.sqrt(num)
  end
end
```

DEMO

https://github.com/epsanchezma/elixir-contracts

FURTHER WORK

- Generate test-cases from Contracts
- Add configuration options to turn-on/off contracts in development and production
- Generate automated documentation from contracts
- Generate QuickCheck tests

TO CONCLUDE

- Design by contract does not replace regular testing strategies
- Contracts add an extra grade of reliability
- It’s not a silver bullet

REFERENCES

- Ariane's case: http://se.inf.ethz.ch/~meyer/publications/computer/ariane.pdf
- DbC History: http://c2.com/cgi/wiki?DesignByContract
- Hoare Logic: https://www.cs.cmu.edu/~aldrich/courses/654-sp07/slides/7-hoare.pdf
- DbC: http://ansymore.uantwerpen.be/system/files/uploads/courses/SE3BAC/06DesignContract.pdf, http://web.cse.ohio-state.edu/software/2221/web-sw1/extras/slides/09.Design-by-Contract.pdf
- Examples: https://www.eiffel.com/


DANKE SCHÖN