BEYOND CONTRACTS

Clojure/West :: March 18, 2013

Paul deGrandis
paul.degrandis@gmail.com

@ohpauleez

MOTIVATION
overview

- Requirements are hard
- Our tools are failing us
- We can do better
- A year of ideas in thirty minutes

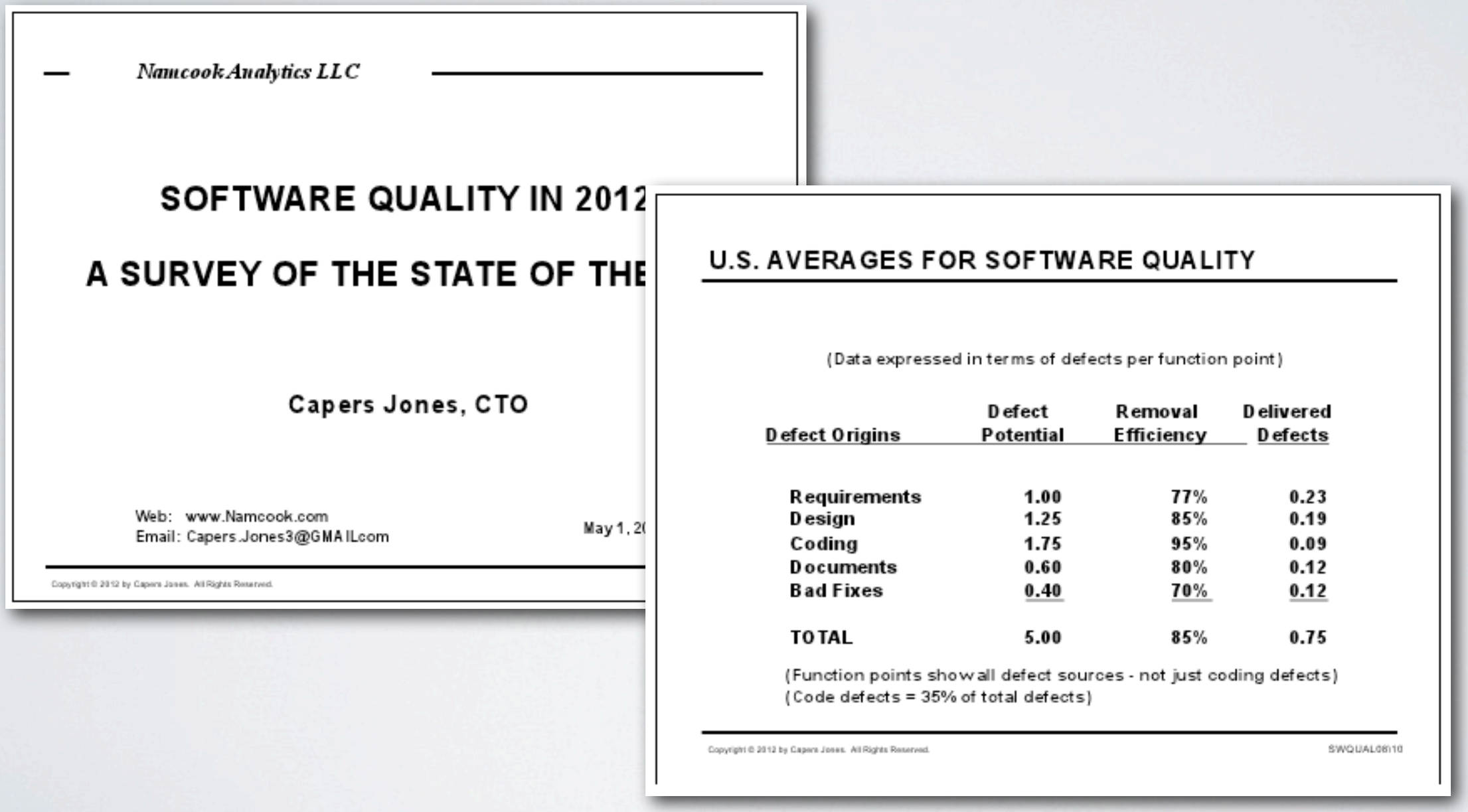
A STORY

VALIDATION - VERIFICATION

- Did we build the right thing?
- Did we build the thing right/correctly?

REQUIREMENTS ARE HARD

you're not alone



Namcook Analytics LLC

SOFTWARE QUALITY IN 2012

A SURVEY OF THE STATE OF THE

Capers Jones, CTO

Web: www.Namcook.com
Email: Capers.Jones3@GMAILcom

May 1, 20

Copyright © 2012 by Capers Jones. All Rights Reserved.

U.S. AVERAGES FOR SOFTWARE QUALITY

(Data expressed in terms of defects per function point)

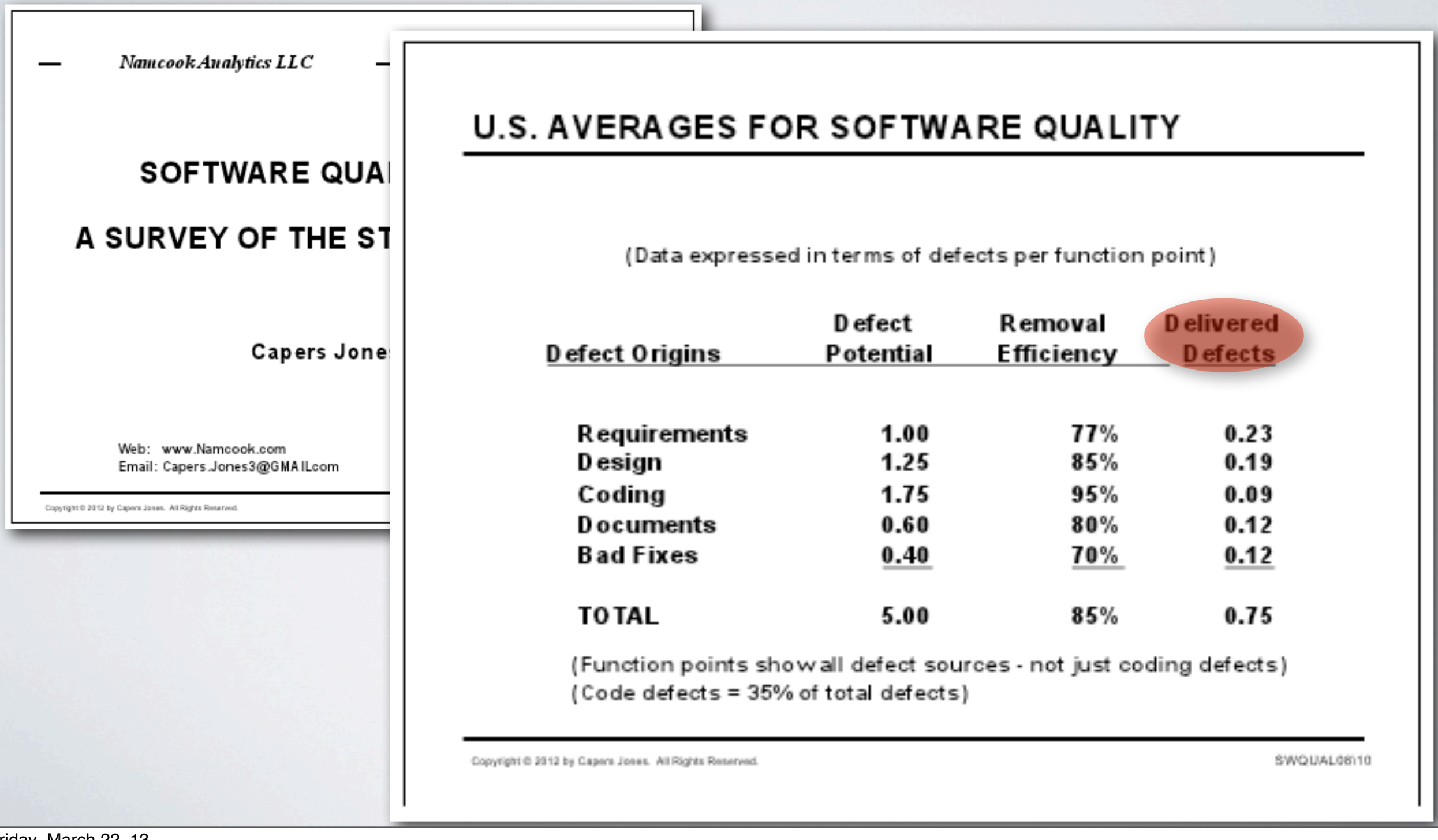
Defect Origins	Defect Potential	Removal Efficiency	Delivered Defects
Requirements	1.00	77%	0.23
Design	1.25	85%	0.19
Coding	1.75	95%	0.09
Documents	0.60	80%	0.12
Bad Fixes	0.40	70%	0.12
TOTAL	5.00	85%	0.75

(Function points show all defect sources - not just coding defects)
(Code defects = 35% of total defects)

Copyright © 2012 by Capers Jones. All Rights Reserved. SWQUAL08\10

REQUIREMENTS ARE HARD

you're not alone



— *Namcook Analytics LLC* —

SOFTWARE QUA

A SURVEY OF THE ST

Capers Jones

Web: www.Namcook.com
Email: Capers.Jones3@GMAIL.com

U.S. AVERAGES FOR SOFTWARE QUALITY

(Data expressed in terms of defects per function point)

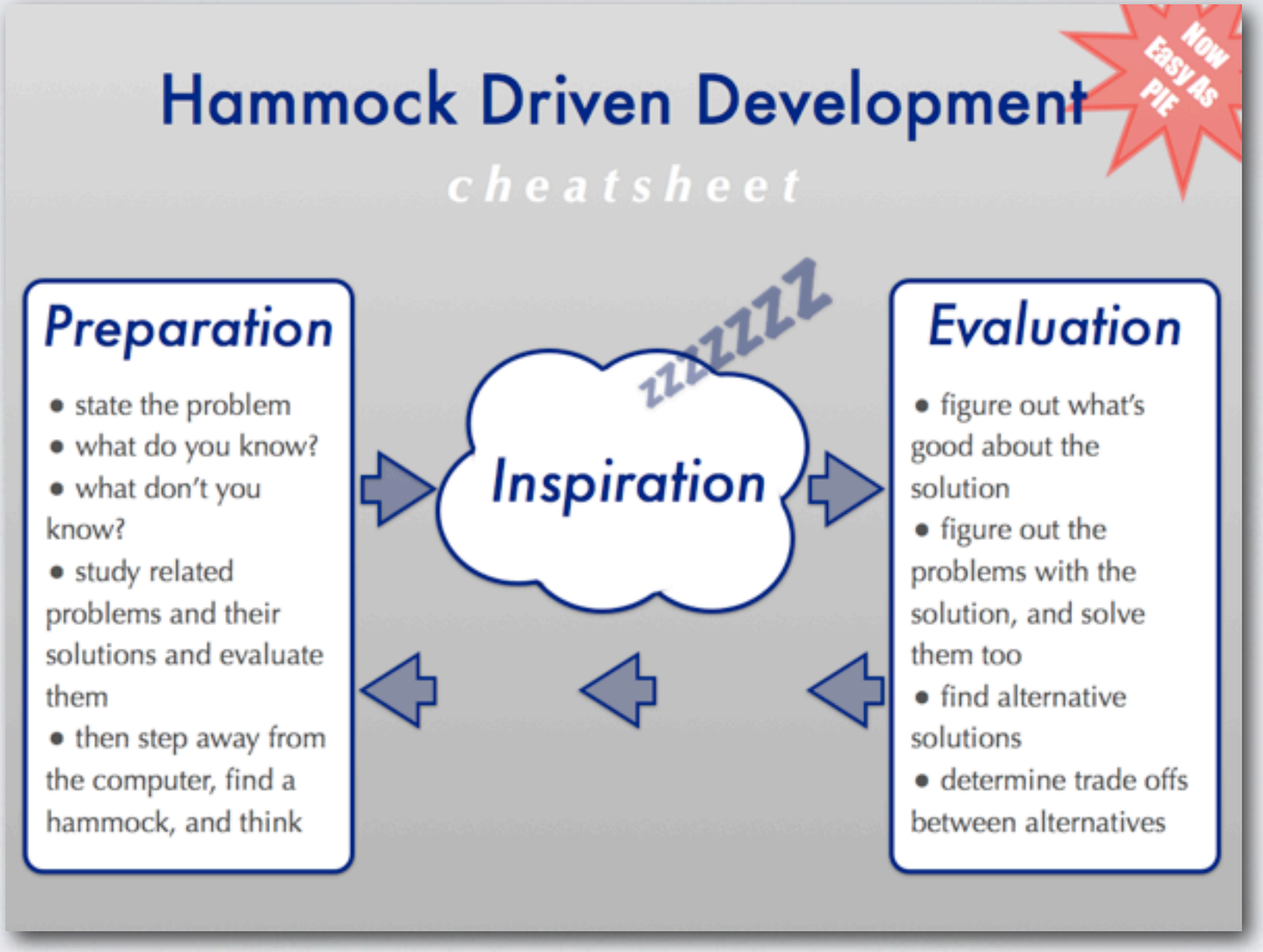
Defect Origins	Defect Potential	Removal Efficiency	Delivered Defects
Requirements	1.00	77%	0.23
Design	1.25	85%	0.19
Coding	1.75	95%	0.09
Documents	0.60	80%	0.12
Bad Fixes	0.40	70%	0.12
TOTAL	5.00	85%	0.75

(Function points show all defect sources - not just coding defects)
(Code defects = 35% of total defects)

SWQUAL08110

REQUIREMENTS ARE HARD

and we know you did your best



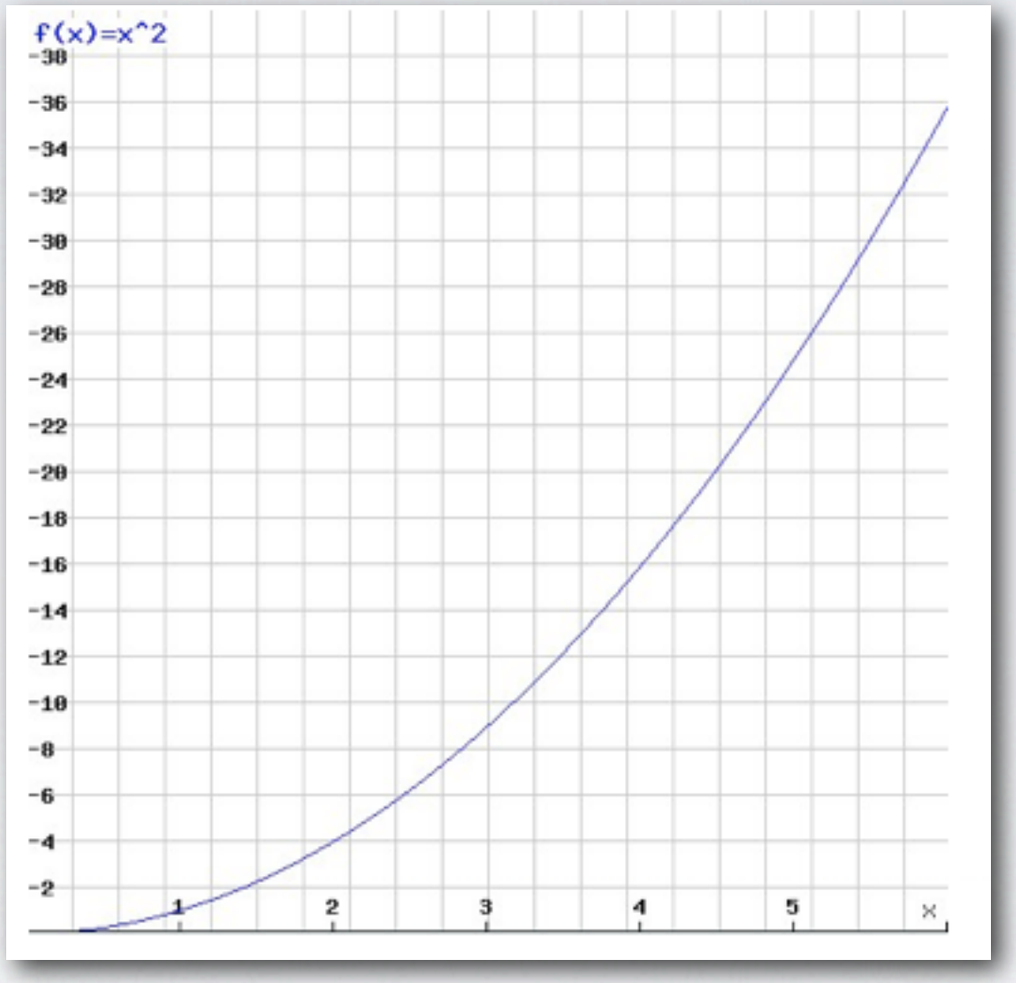
SPECIFICATIONS

an experiment :: guess my function

- 0 → 0
- 1 → 1
- 2 → 4
- 3 → 9
- 4 → 16

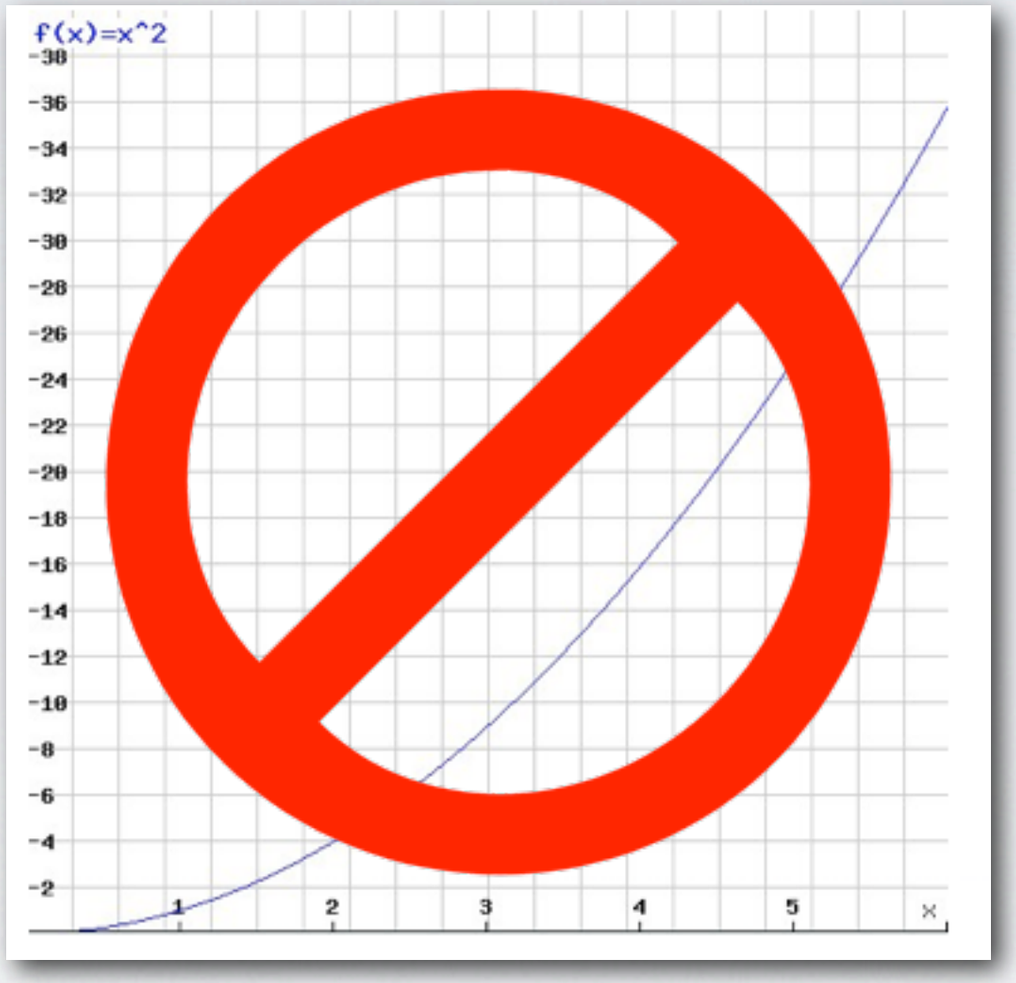
SPECIFICATIONS

an experiment :: guess my function



SPECIFICATIONS

an experiment :: guess my function

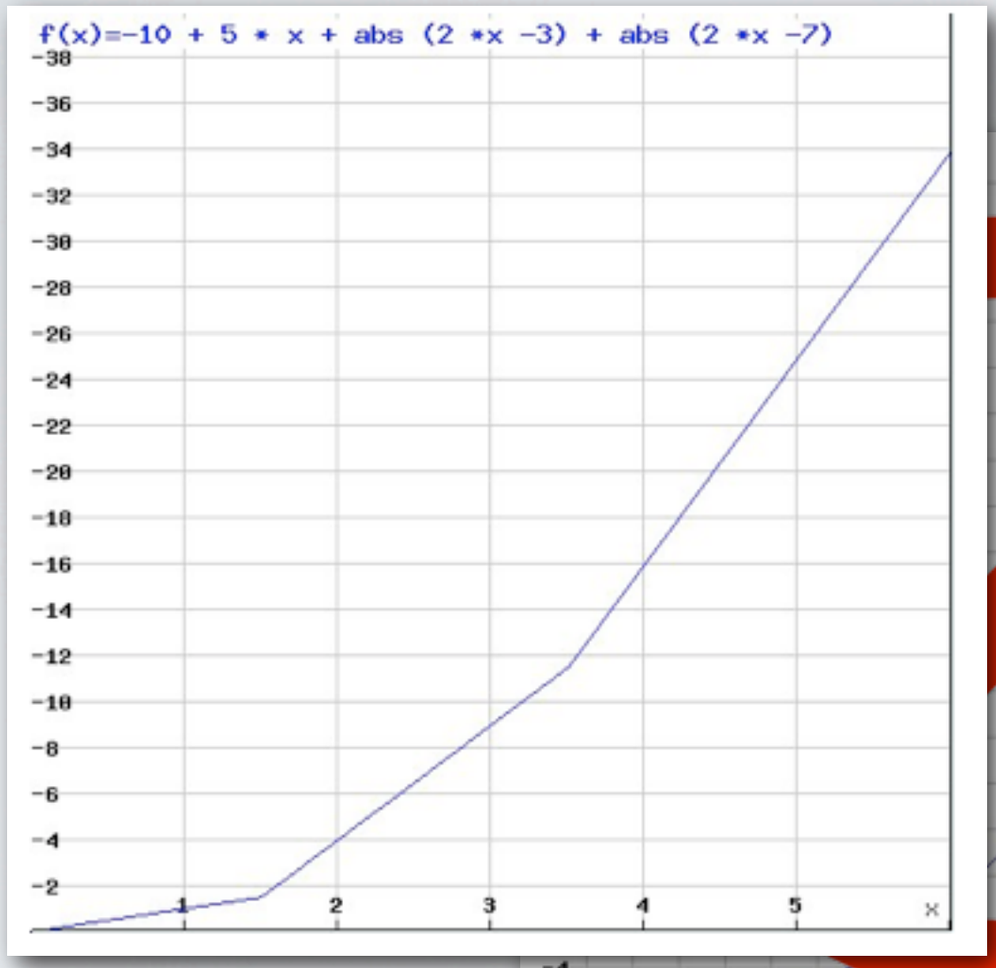


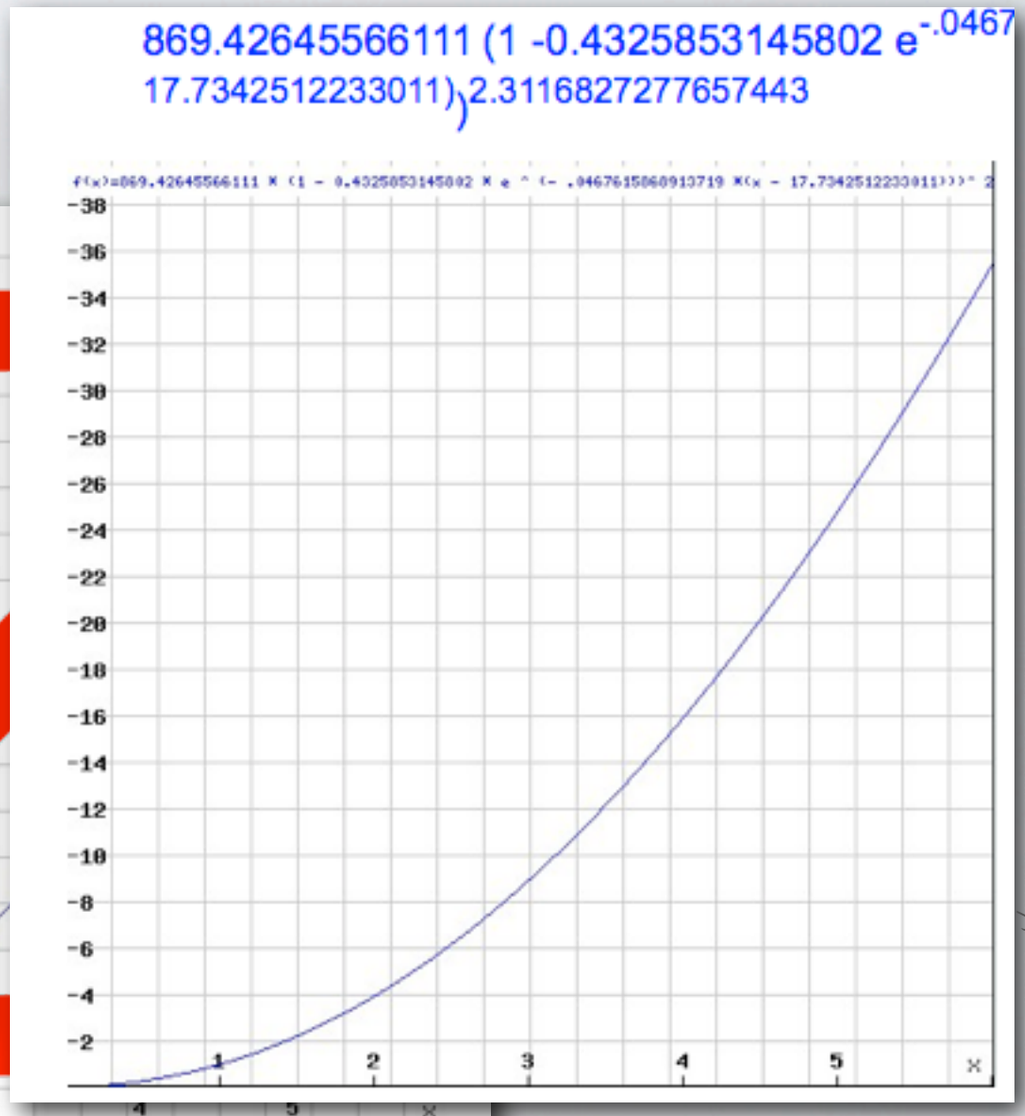
SPECIFICATIONS

an experiment :: guess my function

SPECIFICATIONS

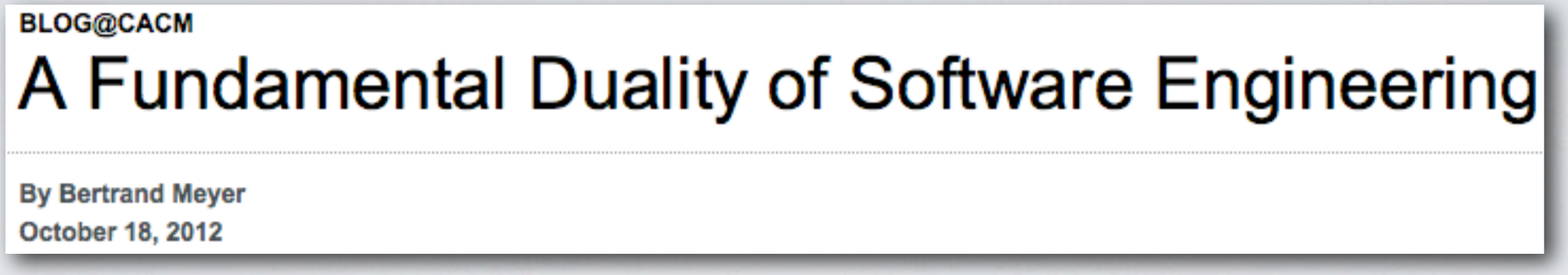
an experiment :: guess my function





ACM - A Fundamental Duality; Oct 2012

SPECIFICATIONS
a fundamental duality


BLOG@CACM

A Fundamental Duality of Software Engineering

By Bertrand Meyer
October 18, 2012

- Specification vs Test

- Test - the program is correct for some values

- Specification - all values for which a program is operational

SPECIFICATIONS
a fundamental duality

BLOG@CACM

A Fundamental Duality of Software Engineering

By Bertrand Meyer
October 18, 2012

"No number of experiments will *prove* a theory"

SPECIFICATIONS

what are specifications

A working definition:

- An operational description of a function or type/record
- Operations bounded by constraints
  - Function: inputs/outputs
  - Record: member invariants

SPECIFICATIONS

our tools are islands

Unit testing?

SPECIFICATIONS

our tools are islands

clojure.test

SPECIFICATIONS

our tools are islands

clojure.test

Generative testing?

SPECIFICATIONS

our tools are islands

clojure.test

test.generative

SPECIFICATIONS

our tools are islands

clojure.test

test.generative

Contracts?!?!

SPECIFICATIONS

our tools are islands

clojure.test

test.generative

core.contracts

SPECIFICATIONS

our tools are islands

clojure.test

test.generative

core.contracts

Typing!!!

SPECIFICATIONS

our tools are islands

clojure.test

test.generative

core.contracts

core.typed

SPECIFICATIONS

our tools are islands

clojure.test

test.generative

core.contracts

core.typed

Query our programs?

SPECIFICATIONS

our tools are islands

clojure.test

test.generative

core.contracts

core.typed

core.logic or datalog

SPECIFICATIONS

our tools are islands

clojure.test

test.generative

core.contracts

core.typed

core.logic or datalog

PAUL!!!! Formal Methods!!!!

SPECIFICATIONS

our tools are islands

clojure.test

test.generative

core.contracts

core.typed

core.logic or datalog

Alloy or Z

SPECIFICATIONS

our tools are islands

clojure.test

test.generative

core.contracts

Functional testing from the top?

core.typed

core.logic or datalog

Alloy or Z

SPECIFICATIONS

our tools are islands

clojure.test

test.generative

core.contracts

Cucumber

core.typed

core.logic or datalog

Alloy or Z

SPECIFICATIONS

our tools are islands

ecowatch.org

WE CAN DO BETTER

- A composable, open system
- Common, unified backend
- À la carte integration
- Proactively part of experimentation and exploration
- Integrated at all phases of software development lifecycle
- Maximize utility, minimize manual effort

WE CAN DO BETTER

One spec provides:

- Contracts
- Generative tests
- Typing information
- Model-checking via Alloy
- Example usage
- Enhanced documentation

WE CAN DO BETTER

it's been done before

- Eiffel and AutoTest
- Haskell and quickcheck
- Alloy + DynAlloy
- Ciao and voluntary assertions

CORE.SPECS

no, it's not a real contrib

```
(spec/defspec tight-inc
    inc
    "An inc that only works for inputs 0-49"
    [^{:uniform [0 50] :enforced true} x]
    :constraints [["The output will always be less than or equal to 50"
                   => (<= % 50)]]
    :typed [[Number -> Number]])
```

CORE.SPECS

no, it's not a real contrib

item being spec'd

a name

doc string

```
(spec/defspec tight-inc
    inc
    "An inc that only works for inputs 0-49"
    [^{:uniform [0 50] :enforced true} x]
    :constraints [["The output will always be less than or equal to 50"
                   => (<= % 50)]]
    :typed [[Number -> Number]])
```

an argument list

any additional specifications

CORE.SPECS

... yet?

```
{:clojure.core.specs/type :defspec,
 :clojure.core.specs/f inc,
 :clojure.core.specs/args [x],
 :clojure.core.specs/ext {:constraints
                          [{:doc "The output will always be less than or equal to 50",
                            :pre nil, :post (<= % 50)}
                           {:doc "The input must be between 0-49",
                            :pre (clojure.core.specs/between? 0 clojure.core/<= x clojure.core/< 50)}],
                         :typed [[Number -> Number]]},
:clojure.core.specs/doc "An inc that only works for inputs 0-49"}
```

BEYOND CONTRACTS

Comments, Questions, Concerns

Paul deGrandis
paul.degrandis@gmail.com

@ohpauleez