Addressing the Regression Test Problem with Change Impact Analysis for Ada

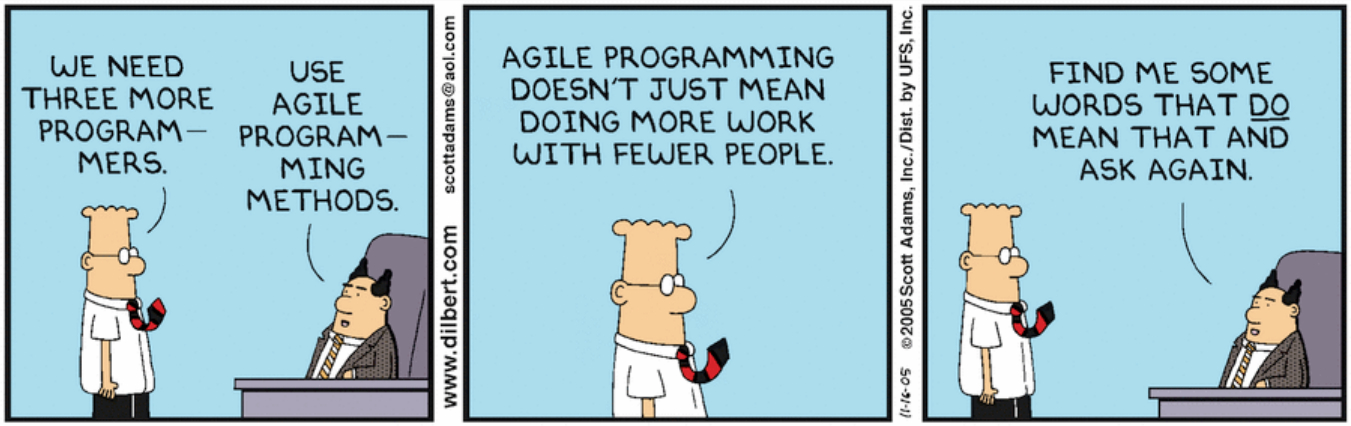
Andrew V. Jones, Vector Software, Inc.

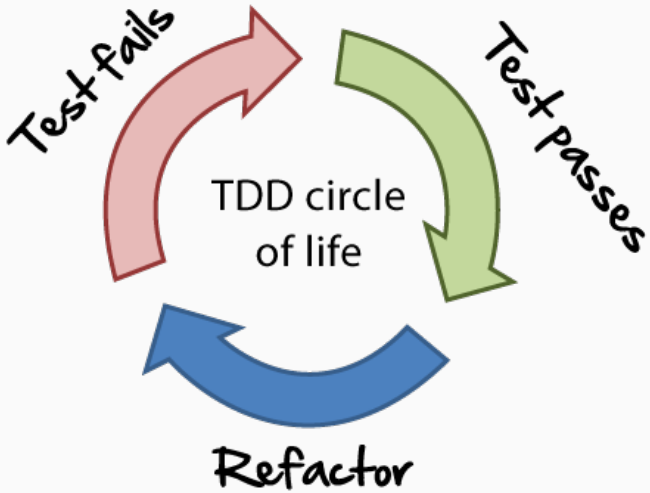
Ada-Europe 2016, Pisa

VECTOR software

Setting the scene

I'm a "cool" developer ...

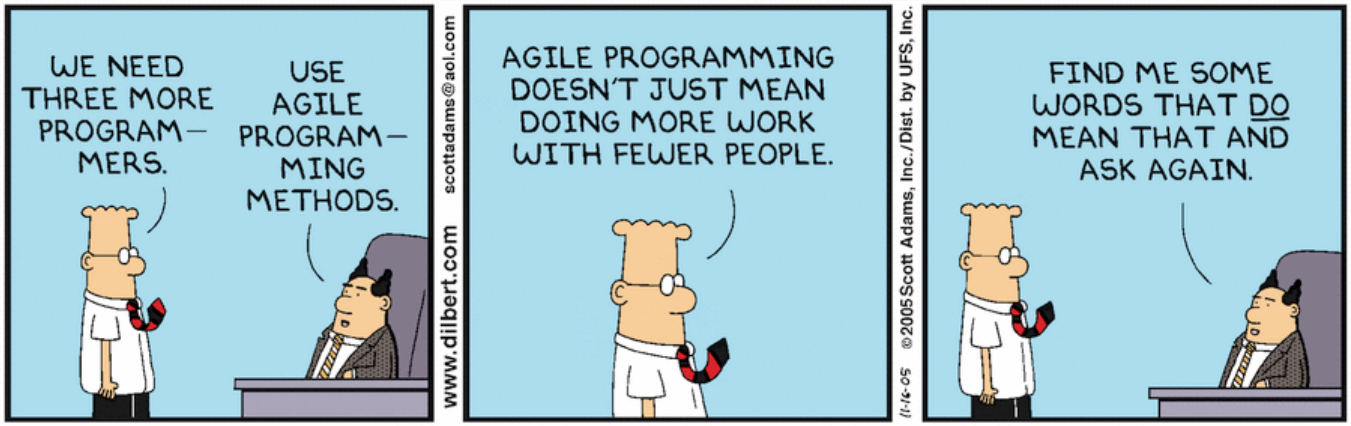


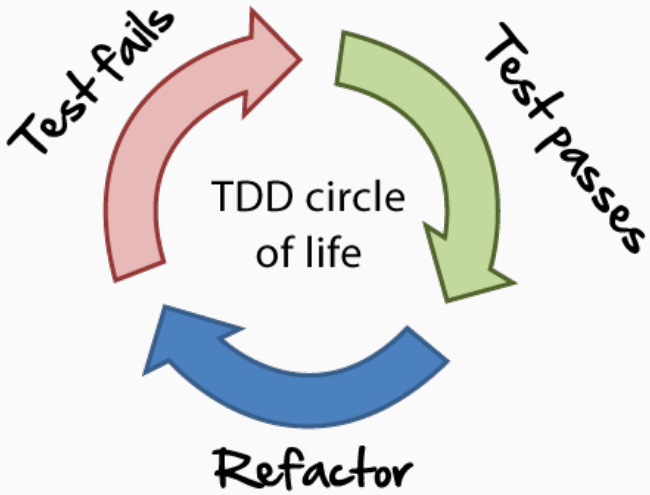


VECTOR software

Setting the scene

I'm a "cool" developer ...





VECTOR software

The problem

- I’ve made a change to my software …
- and I now need to test it …
- but I have thousands of tests …

Help!

VECTOR software

The solution

The *test-case selection problem* [2]:

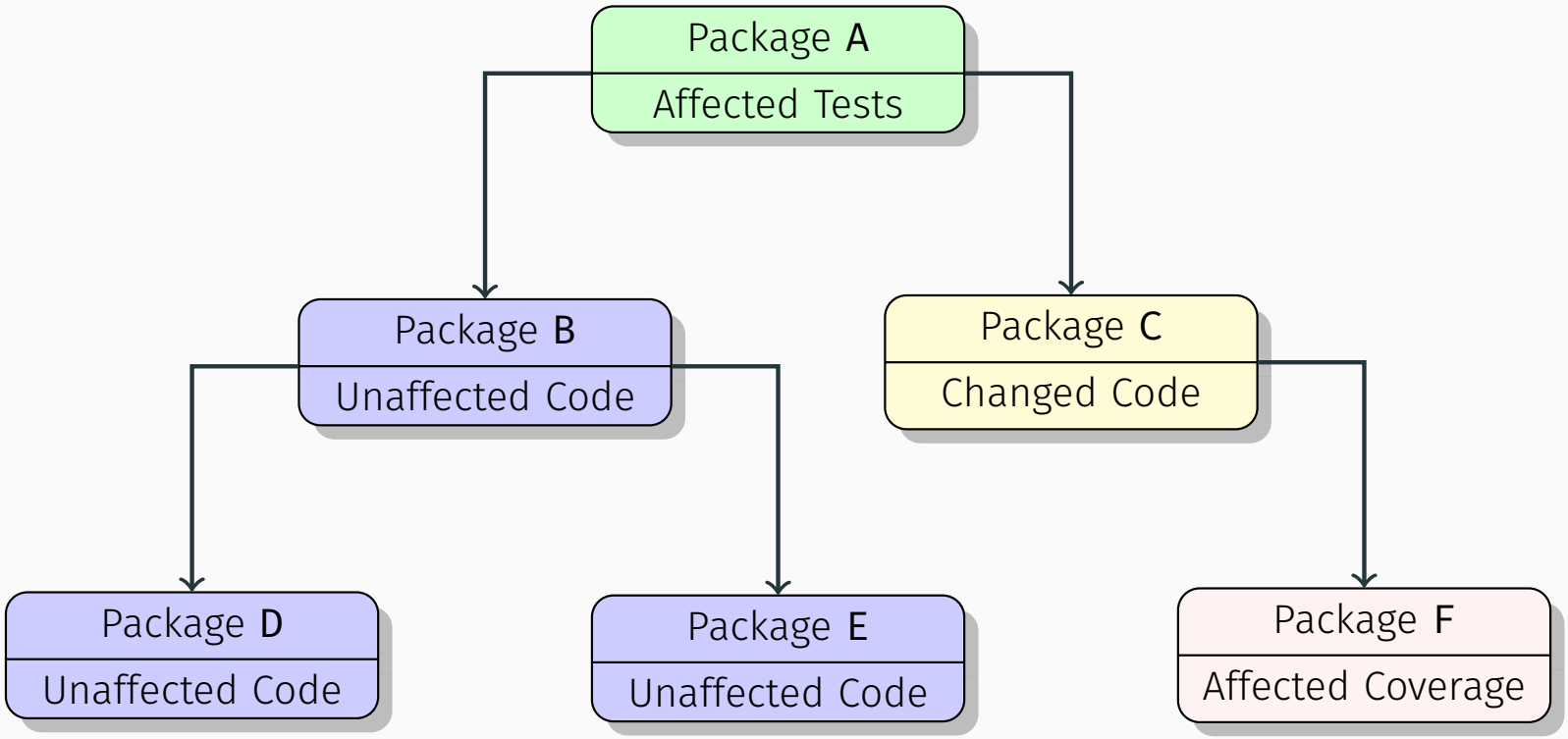
> *"determine which test-cases need to be re-executed [...] in order to verify the behaviour of modified software"*

With a focus on:

- Minimality
- Ada

VECTOR software

Pretty picture!



VECTOR software

The intuition

- Construct a DAG $\mathcal{D}$ of the dependencies of the SUT
- Calculate the transitive closure $\mathcal{D}^*$
  - $(x, y) \in \mathcal{D}^*$ has the reading "x depends on y"
- If $(x, y) \in \mathcal{D}^*$ and y has been modified, re-test x!

The key here is how we construct $\mathcal{D}$ ...

VECTOR software

A minor divergence: code coverage

A way of tracking what lines have been executed

Commonly used in:

- TDD
- Safety-critical systems
    - The idea of “test completeness”

VECTOR software

Constructing the dependency graph

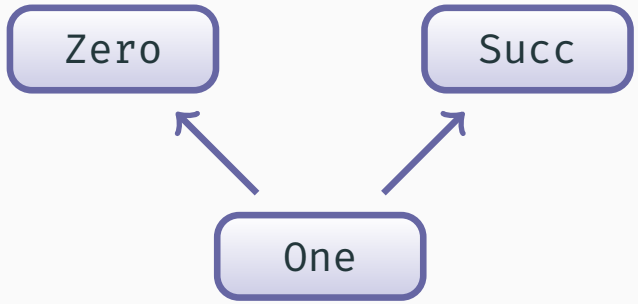
- Statically – anything outside of a function/procedure
  1. Type and Ada specification dependencies – A with's B as part of A's spec
  2. Uses and Ada body dependencies – A with's B as part of A's body

- Dynamically – anything inside of a function/procedure
  3. Subprogram invocation and coupling – `Foo` calls `Bar`

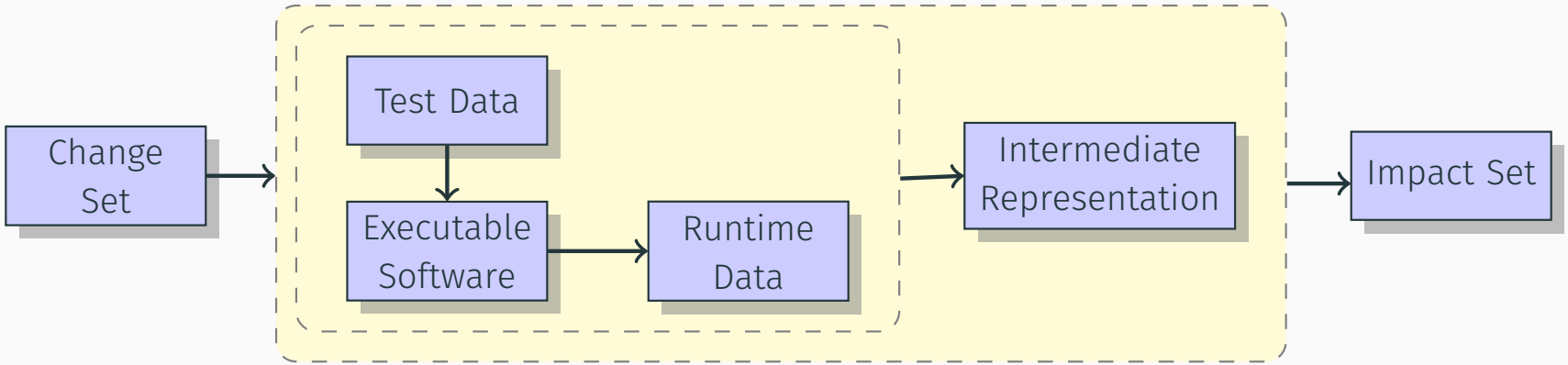
VECTOR software

Dynamic dependencies

```
1  package body Peano is
2
3     function One return Integer is
4     begin
5         return Succ(Zero);
6     end One;
7
8     function Zero return Integer is
9     begin
10       return 0;
11    end Zero;
12
13    function Succ (Val : in Integer)
14        return Integer is
15    begin
16        return Val + 1;
17    end Succ;
18
19 end Peano;
```



Dynamic impact analysis



VECTOR software

A larger example

```
1  package A is
2
3      function Foo
4          return Integer;
5
6  end A;
```

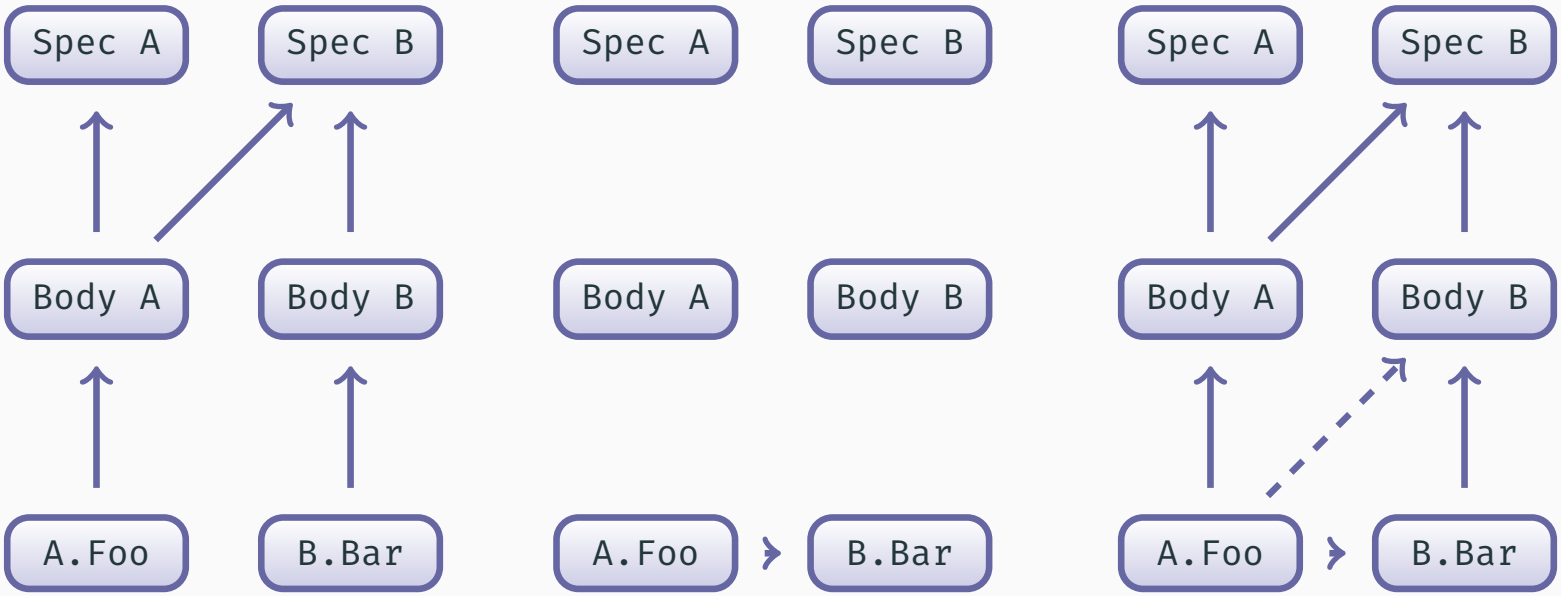
```
1  package B is
2
3      function Bar
4          return Integer;
5
6  end B;
```

```
1  with B;
2
3  package body A is
4
5      Qux : Integer;
6
7      function Foo return Integer
           is
8      begin
9          return Qux + B.Bar;
10     end;
11
12 begin
13
14     Qux := 0;
15
16 end A;
```

```
1  package body B is
2
3      Narf : Integer;
4
5      function Bar return Integer
           is
6      begin
7          return Narf;
8      end;
9
10 begin
11
12     Narf := 0;
13
14 end B;
```

VECTOR software

Combining static and dynamic



(a) Static Dep. (b) Dynamic Dep. (c) Combined Dep.

Intermediate representation

$$\mathit{Contains} : \mathit{Package} \rightarrow \mathit{Subprogram}^*$$

$$\mathit{Uses} : \mathit{Package} \times \{\mathit{Body}, \mathit{Spec}\} \rightarrow \mathit{Package}^*$$

$$\mathit{Covers} : \mathit{Test} \rightarrow \mathit{Subprogram}^*$$

VECTOR software

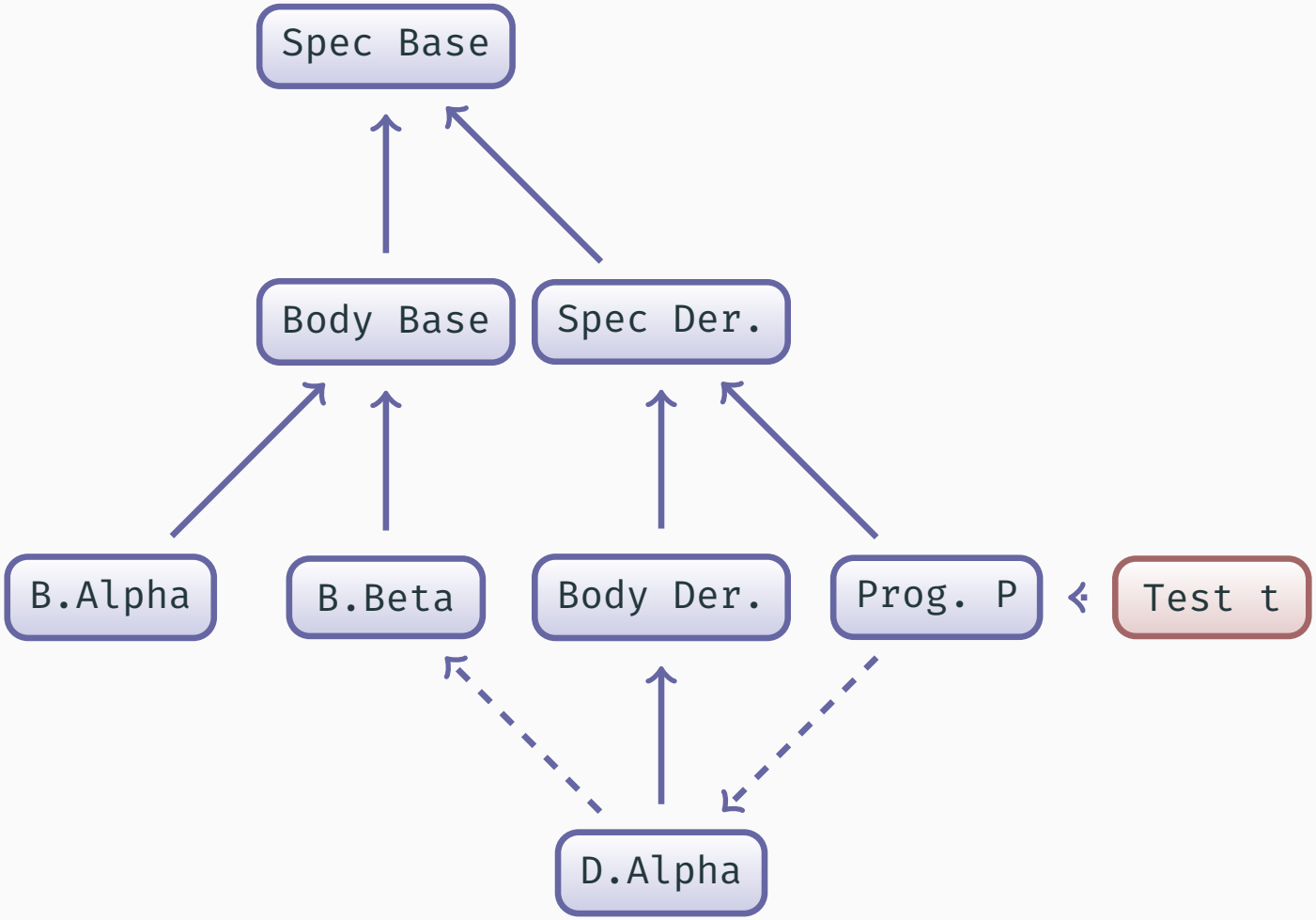
Calculating the selection

Simplistic approach

1. Use a work-list to calculate $\mathcal{D}^*$ at the subprogram level
2. Find all tests with coverage on the affected subprograms
3. Re-execute those tests

VECTOR software

What about OOP?



Consider: `Derived` now contains `Beta`, or a change to `Base`'s body.

VECTOR software

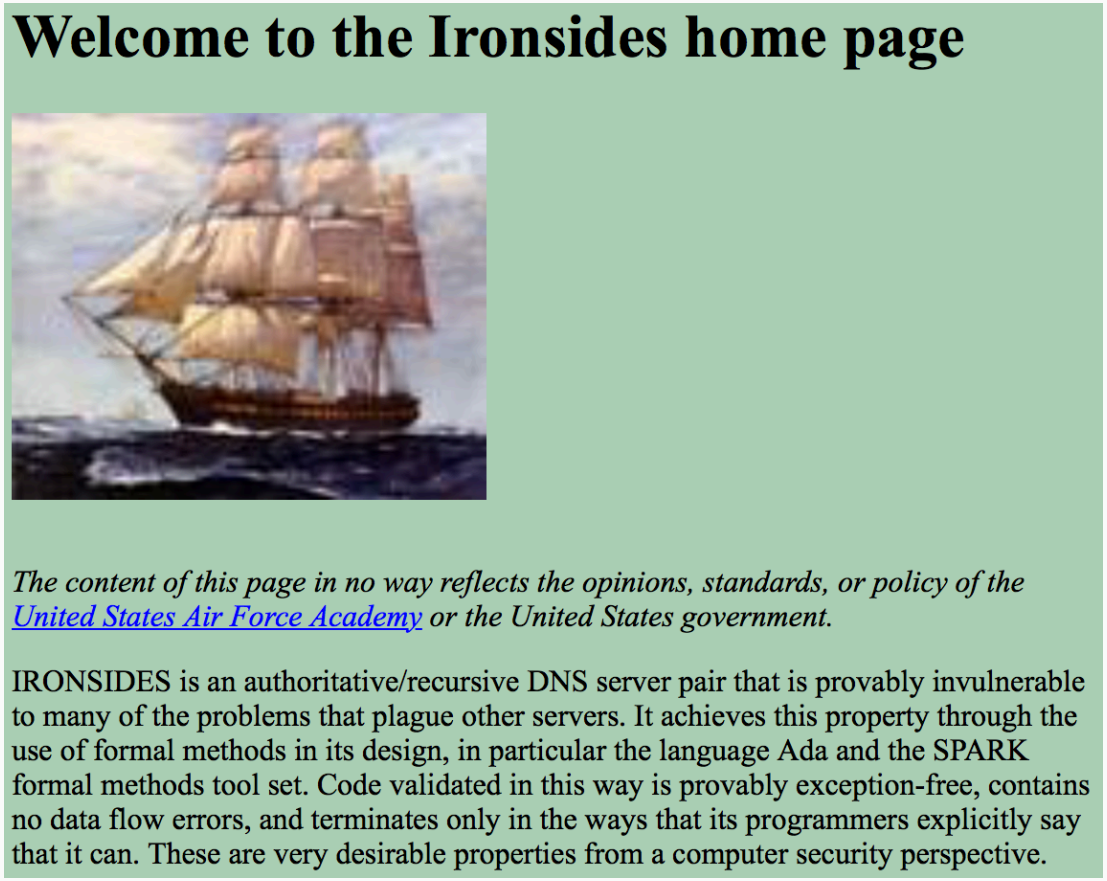
Implementation

Not a demo, not a sales pitch: functionality implemented inside of the commercial unit-testing tool VectorCAST.

VECTOR software

Evaluation

We took two open-source code bases ...



Higher performance than `bind`!!! See: [1].

VECTOR software

Evaluation

Build VectorCAST environments on them, and automatically created tests:

Metric	Malaise	IRONSIDES
Number of files	9	9
Number of lines	654	4,745
Number of non-empty Ada lines	468	3,441
Number of subprograms	46	97
Aggregate complexity metric [4]	94	492
Total number of tests	228	573
Coverage (statement / branch)	68% / 68%	47% / 36%

Next we automatically modified the source-code ...

VECTOR software

Automated change

Simplistic (but real!) example taken from [3]:

```
1  function Is_Delimiter (C : Character) return Boolean is
2  begin
3      null;
4      case C is
5          when '&' | ''' | '(' | ')' | '*' | '+' |
6               ',' | '-' | '.' | '/' | ':' | ';' |
7               '<' | '=' | '>' | '|' =>
8              return True;
9          when others =>
10             return False;
11     end case;
12 end Is_Delimiter;
```

And then we re-ran the test-suite ...

VECTOR software

Experimental results

Example	Mode	Units Changed	Subprograms Changed	# Tests Executed	Build + Exec. Time (s)
Malaise	Without CBT With CBT	9	21	4,788 165	1,002.48 165.85
IRONSIDES	Without CBT With CBT	9	93	53,289 1,347	6,986.17 1,147.14

Take-home

- 97% reduction in tests executed
- 84% reduction in time spent testing

  $\Rightarrow$ Re-world testing should scale better

VECTOR software

Artefacts

All code, VectorCAST artefacts and the test harness are available under a MIT license:

`https://github.com/andrewvaughanj/CBT_for_Ada_Examples`

VECTOR software

What's next?

Right now, the approach is safe but still quite coarse:

- Selects *at least* what is necessary, but is not minimal

We have a number of ideas in this area:

- Changes only affecting certain branches (e.g., constrained to one branch of an if statement)
- Uses of package-level variables
- Innocuous changes (e.g., a new variable, new procedure)

VECTOR software

Summing it up

- Effective approach for “change-based testing” of Ada
- Can dramatically reduce the re-testing effort (97% reduction on a real-world examples!)
- Designed to speed-up developer testing; shouldn’t replace complete end-to-end runs
- Available now in VectorCAST!

VECTOR software

Fin.

Questions?

Looking for collaborations between Vector and academia – speak to me if this is interesting:

andrew.jones@vectorcast.com

VECTOR software

References I

[1] M. C. Carlisle. **IRONSIDES homepage.** `http://ironsides.martincarlisle.com`, April 2015. Accessed: 2016-03-22.

[2] E. Engström, P. Runeson, and M. Skoglund. **A Systematic Review On Regression Test Selection Techniques.** *Information & Software Technology*, 52(1):14–30, 2010.

[3] P. Malaise. **PMA's Ada contrib.** `http://pmahome.cigecompta.fr/ada/html/REPOSIT_LIST.html`, January 2016. Accessed: 2016-03-22.

[4] A. H. Watson, T. J. McCabe, and D. R. Wallace. **Structured Testing: A Software Testing Methodology Using the Cyclomatic Complexity Metric.** Technical Report Special Publication 500-235, U.S. Department of Commerce/National Institute of Standards and Technology, September 1996.

VECTOR software