Using DSLs for Testing

Pandy Knight
Open Security Summit
January 14, 2021

Andrew “Pandy” Knight

Lead Software Engineer in Test
PrecisionLender, a Q2 Company

AutomationPanda.com
@AutomationPanda

Disclaimer:

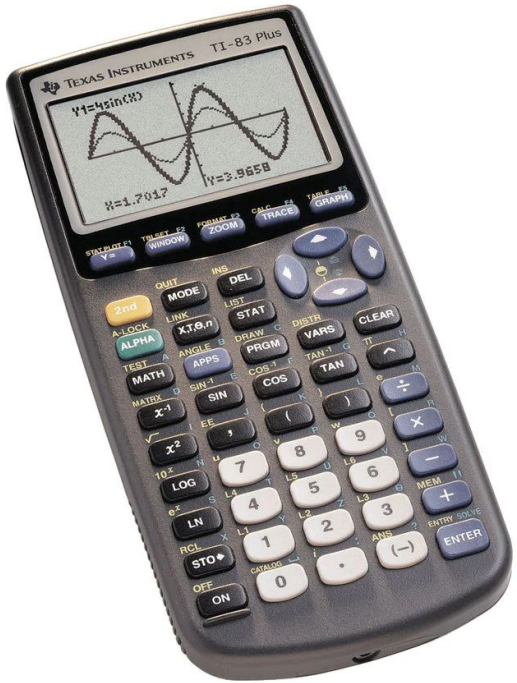
Security

Testing

How can we use
Domain-Specific Languages
for **security testing**?

My Story

High School




```
NORMAL FLOAT AUTO a+bi DEGREE MP
EDIT MENU: [alpha] [f5]
PROGRAM:AA
:■10→N
:If dim(L1)≠dim(L2
:Stop
:ClrDraw
:L1(1→X
:L2(1→Y
:L1(2→S
:L2(2→T
:Line(S,T,X,Y,10,1
```


Java


python


++


COBOL

Languages

```
class CreateAddresses
  def change
    create_table :addresses do |t|
      t.string :street1
      t.string :street2
      t.string :city
      t.string :state
      t.string :zipcode
    end
  end
end
```

College

Language fundamentals:

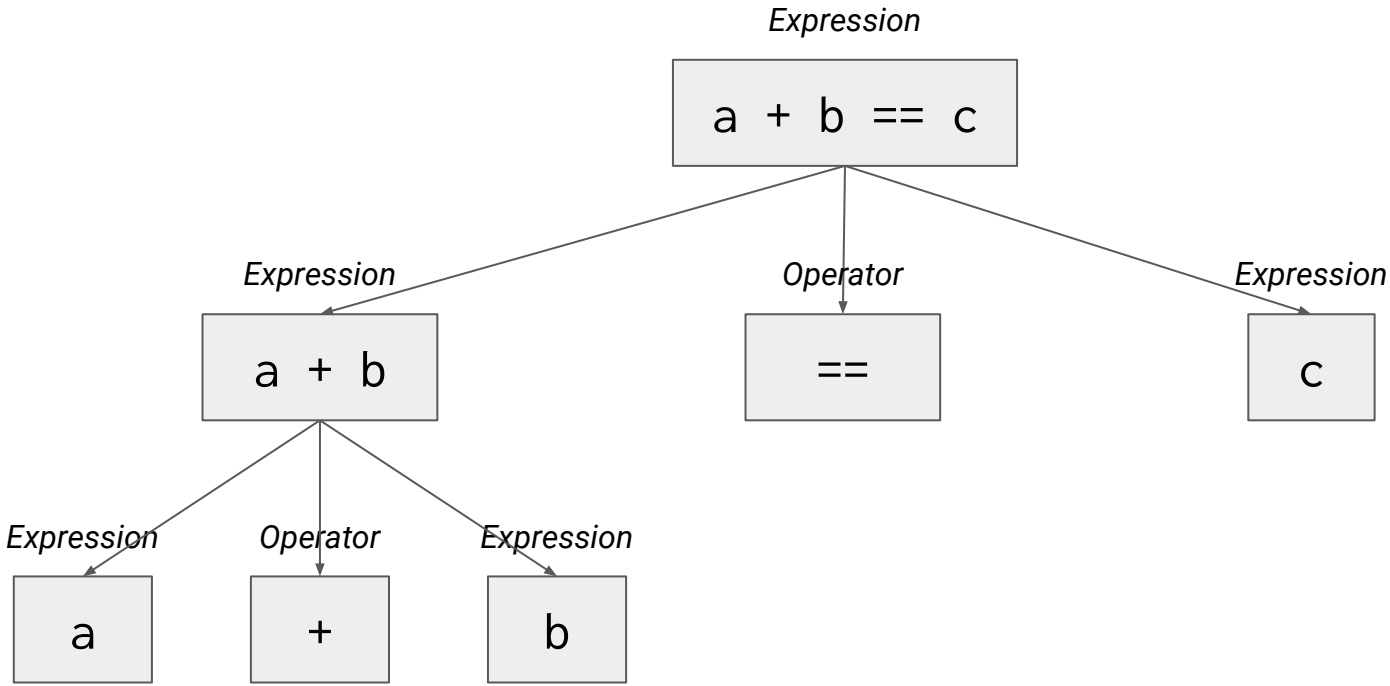
- Computer Science Theory
- Programming Language Concepts
- Functional Programming
- **Compiler Construction**

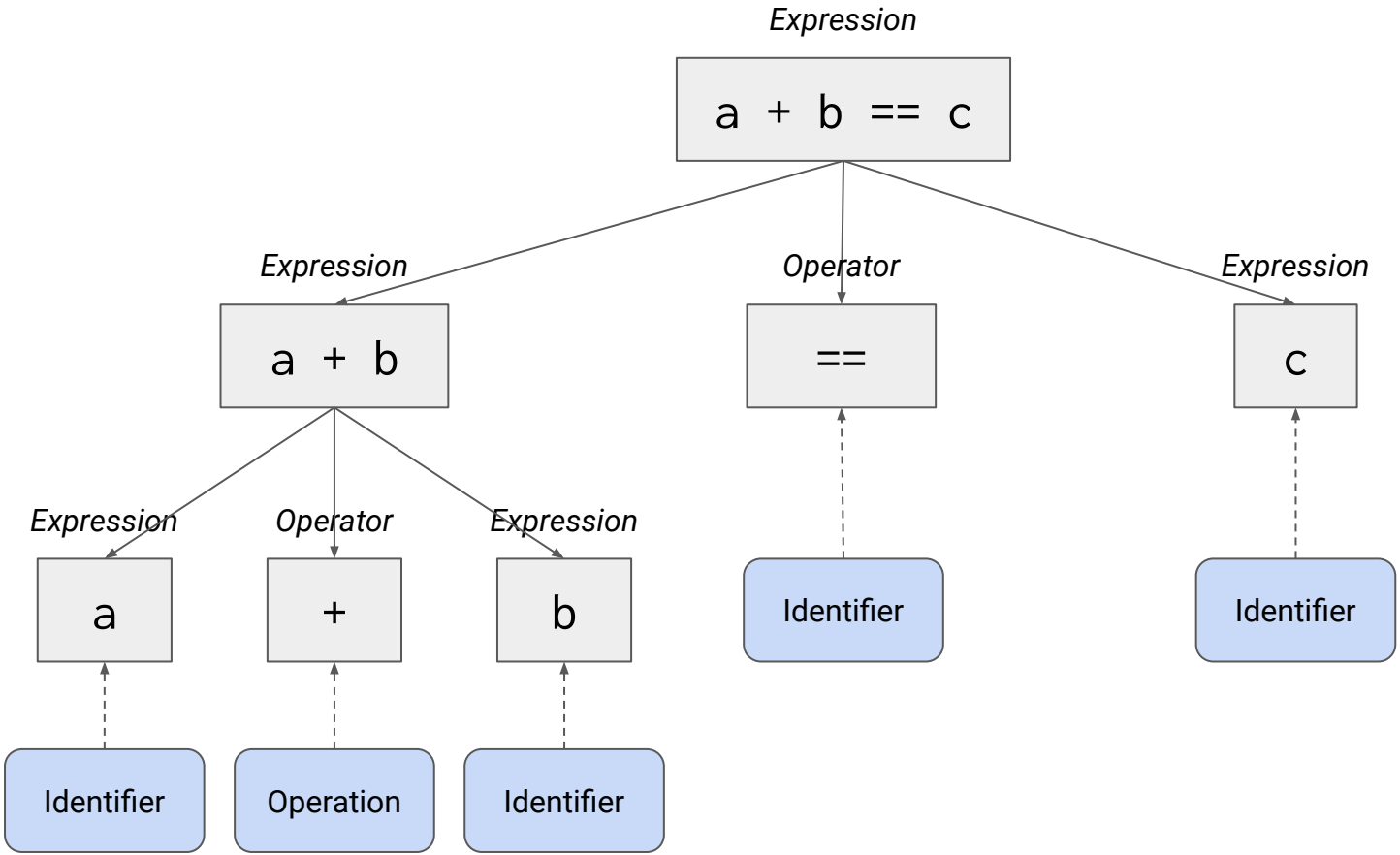
College Lessons Learned

1. **Computer science** is just the study of machine instruction.
2. A **language** is just a tool for writing instructions.
3. An **interpreter** is just a program for executing instructions.
4. A **compiler** is just a program for translating instructions.

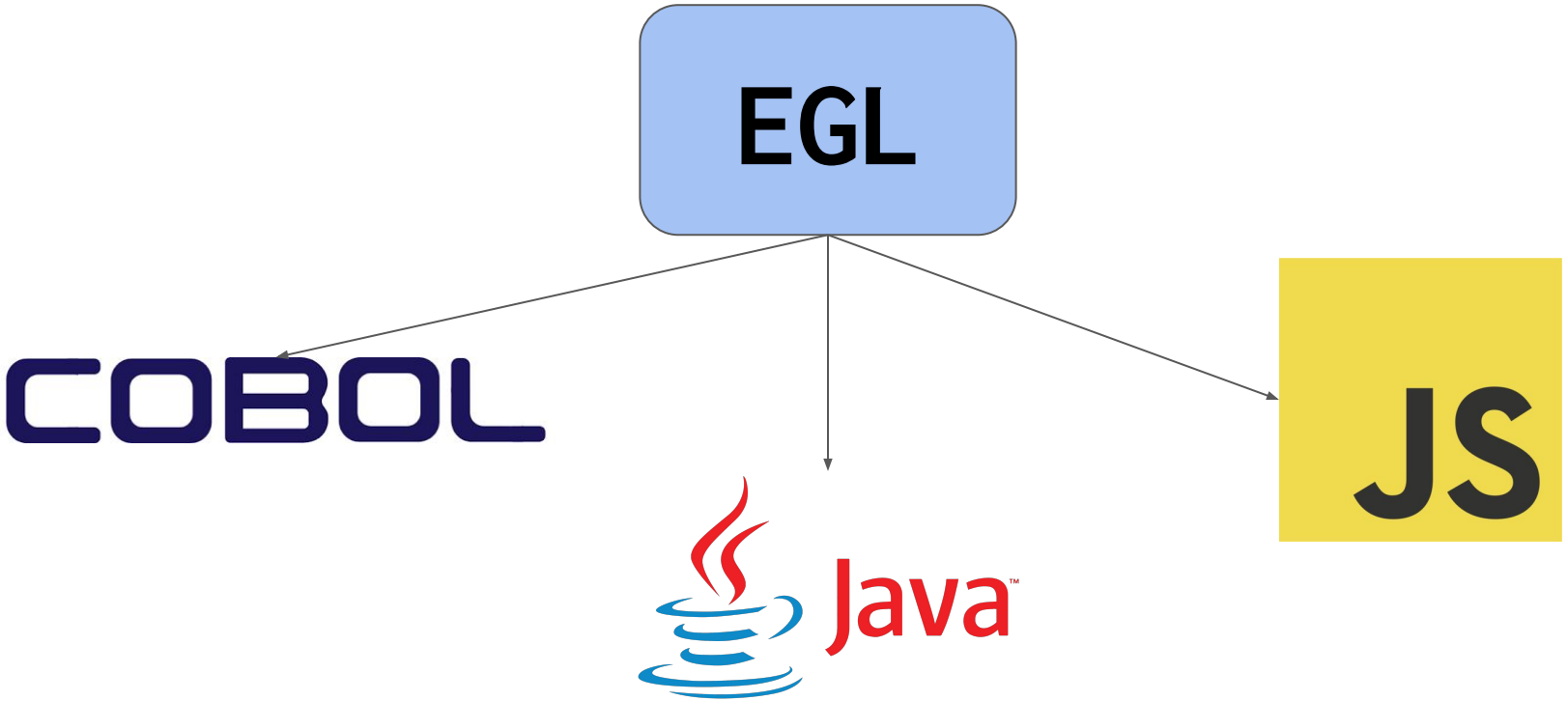
You can make your own language if you make an interpreter/compiler!

```
a + b == c
```


Expression
a + b == c
Expression
a + b
Operator
==
Expression
c
Expression
a
Operator
+
Expression
b


Expression
a + b == c
Expression
a + b
Operator
==
Expression
c
Expression
a
Operator
+
Expression
b
Identifier
Identifier
Identifier
Operation
Identifier

IBM Internship



The purpose of a language is to **make instructions easy to write**.

NetApp Test Automation

Background:

- NetApp hired me onto the Platform QA team
- My primary job was test automation
- NetApp did all test automation in Perl

First project:

- The Bangalore team runs Perl test scripts every night
- Get those tests running in our Raleigh lab!

The code **didn’t work.**

Test Automation Problems

1. Simple test cases needed lots of code
2. Code duplication was rampant
3. There were no coding standards or conformity in style
4. Tests were out of date, and updates were difficult to make
5. Test code was not managed using version control

...Not to mention awkward organizational politics

I fell down the test automation rabbit hole.

The Goal

Test Case

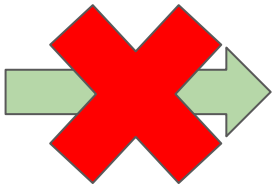
1. -
2. -
3. -
4. -
5. -

```
sub calc_srs {
    my $tptr = shift;
    for ( keys %$tptr ) {
        $tptr->{$_}{mov} = $tptr->{$_}{point_spread}/$tptr->{$_}{games_played};
        $tptr->{$_}{srs} = $tptr->{$_}{mov};
        $tptr->{$_}{sos} = 0;
    }
    my $delta = 10.0;
    my $iters = 0;
    while ( $delta > 0.001 ) {
        $iters++;
        $delta = 0.0;
        for ( keys %$tptr ) {
            my $sos = 0.0;
            for my $g ( @{$tptr->{$_}{played}} ) {
                $sos += $tptr->{$g}{srs};
            }
            $sos /= $tptr->{$_}{games_played};
            $tptr->{$_}{sos} = $sos;
            $tptr->{$_}{oldsrs} = $tptr->{$_}{srs};
            $tptr->{$_}{srs} = $tptr->{$_}{mov} + $sos;
            my $newdelt = abs( $tptr->{$_}{srs} - $tptr->{$_}{oldsrs} );
            $delta = max ( $newdelt, $delta );
        }
    }
    print "iters = $iters\n" if $debug;
    return;
}
```

The Problems with Perl

Test Case

1. -
2. -
3. -
4. -
5. -




```
sub calc_srs {
    my $tptr = shift;
    for ( keys %$tptr ) {
        $tptr->{$_}{mov} = $tptr->{$_}{point_spread}/$tptr->{$_}{games_played};
        $tptr->{$_}{srs} = $tptr->{$_}{mov};
        $tptr->{$_}{sos} = 0;
    }
    my $delta = 10.0;
    my $iters = 0;
    while ( $delta > 0.001 ) {
        $iters++;
        $delta = 0.0;
        for ( keys %$tptr ) {
            my $sos = 0.0;
            for my $g ( @{$tptr->{$_}{played}} ) {
                $sos += $tptr->{$g}{srs};
            }
            $sos /= $tptr->{$_}{games_played};
            $tptr->{$_}{sos} = $sos;
            $tptr->{$_}{oldsrs} = $tptr->{$_}{srs};
            $tptr->{$_}{srs} = $tptr->{$_}{mov} + $sos;
            my $newdelt = abs( $tptr->{$_}{srs} - $tptr->{$_}{oldsrs} );
            $delta = max ( $newdelt, $delta );
        }
    }
    print "iters = $iters\n" if $debug;
    return;
}
```

What if I could make a
better language for testing?

Domain-Specific Languages

A *general-purpose* language can write software for any need:

- Python, Java, JavaScript, C, C++, C#, Perl, etc.

A *domain-specific* language (DSL) focuses on the needs of a particular problem:

- SQL - relational database queries
- HTML - web page markup
- YAML - text format for object specification

DS

I created a DSL named **DS** for NetApp tests.

- “DS” was short for “Design Steps”
- Written in “.ds” text files
- Executed by an interpreter written in Perl using Parse::RecDescent
- Had basic commands for test controls
- Could call modules by name written in Perl to use NetApp’s libraries

Example DS Test

```
PROCEDURE <Reboot SP And Verify> {
    SETUP {
        RUN mod<Store Current SP Version>;
    }
    MAIN {
        RUN mod<Reboot SP From SP CLI>;
        SLEEP 180;
        VERIFY mod<Is SP Online?>;
        VERIFY EQUAL [mod<Booted SP Image>, mod<Stored SP Image>];
        VERIFY EQUAL [mod<Booted SP Version>, mod<Stored SP Version>];
    }
}
```

Advantages of DS

Tests were simple, parts were reusable, and automation was stable!

The DS language automatically handled:

- Assertions
- Reporting results
- Logging
- Parallel execution
- Setup and cleanup test phases
- Automatic retries

500+

Tests written in DS

The Legacy of DS

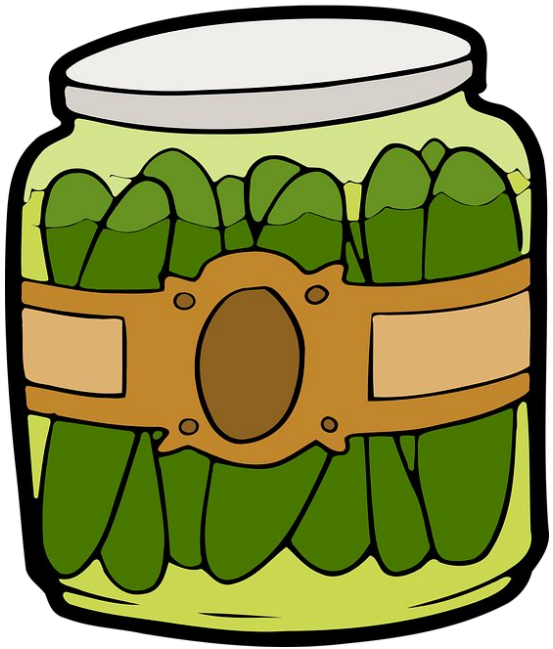
I proved that a test automation DSL works wonderfully.

However, a DSL takes a lot of time, effort, and expertise to build.

After I left NetApp, people could maintain the tests but not the language itself.

Good things

were right around the corner.



BDD

1. Collaboration
2. Automation

Gherkin

```
Scenario: Basic DuckDuckGo Search

  Given the DuckDuckGo home page is displayed

  When the user searches for "panda"

  Then results are shown for "panda"
```

Gherkin Automation

```
When the user searches for "panda"
```



```
@when(parsers.parse('the user searches for "{text}"'))
def search_phrase(browser, text):
    search_input = browser.find_element_by_id('search_form_input_homepage')
    search_input.send_keys(text + Keys.RETURN)
```

What makes **Gherkin** so popular for testing?

BDD provided **Gherkin** as
the first widely-available **DSL**
for test automation.

I’ve built test automation projects with C#, SpecFlow, and Selenium WebDriver.

My current project runs **~15K tests/week**.

How can we use
Domain-Specific Languages
for **security testing**?

Security testing is just a special type of **functional testing**.

Password Safety

```
Scenario: Login page should not reveal which credential is invalid
  Given the login page is displayed
  When the user enters a valid username
  But the user enters an invalid password
  And the user clicks the login button
  Then the login page says credentials were invalid without specifying
which one
```

User Lockout

```
Scenario: Lock the user out after five successive failed login attempts
  Given the login page is displayed
  When the user attempts to login "5" times with invalid passwords
  Then the login page says the user's account is locked out
  And the user cannot successfully log in with the correct password
```

Limited User Access

```
Background:
  Given a non-admin user is logged into the app

Scenario: Non-admin users do not see the Administration menu
  Then the Administration menu is not displayed

Scenario: Non-admin users cannot navigate the Administration menu
  When the user attempts to navigate directly to the Administration URL
  Then the app displays an HTTP 500 error page
  And the error page does not indicate that the Administration page was targeted
```

What other **security tests** could you write?

We can do it!

Security

Testing

Calls to Action

1. Can DSLs make security testing easier and more widely practiced?
2. Is Gherkin good enough, or do we need a DSL specific to security?
3. Could we write a set of “standard” security tests using a DSL?

Andrew “Pandy” Knight

Lead Software Engineer in Test
PrecisionLender, a Q2 Company

AutomationPanda.com
@AutomationPanda