

Lessons learnt from using DSLs for Automated Software Testing TAIC-PART 2015

Mark Micallef, Christian Colombo

PEST Research Lab, University of Malta

Tuesday, April 07, 2015





Product Owner



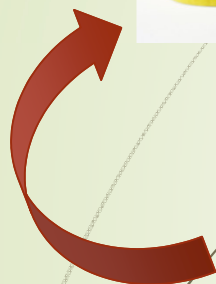
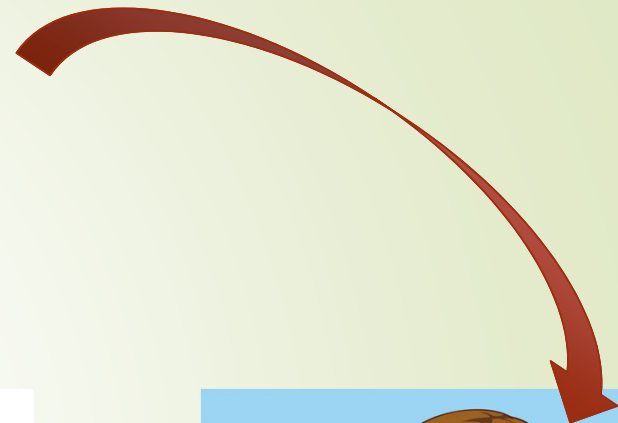
Tester / QA



Project Manager



Developer





Isn't this already being done?

Given I am a premium user

When I place a bet on a football match

And I win the bet

Then I will win 10% more than the advertised odds for the match



Domain Specific Languages (DSLs)

“A computer programming language of limited expressiveness focused on a particular domain”

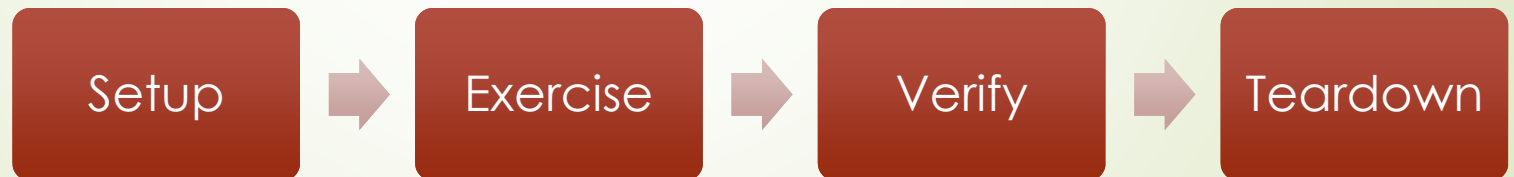
Martin Fowler

Internal vs External DSLs



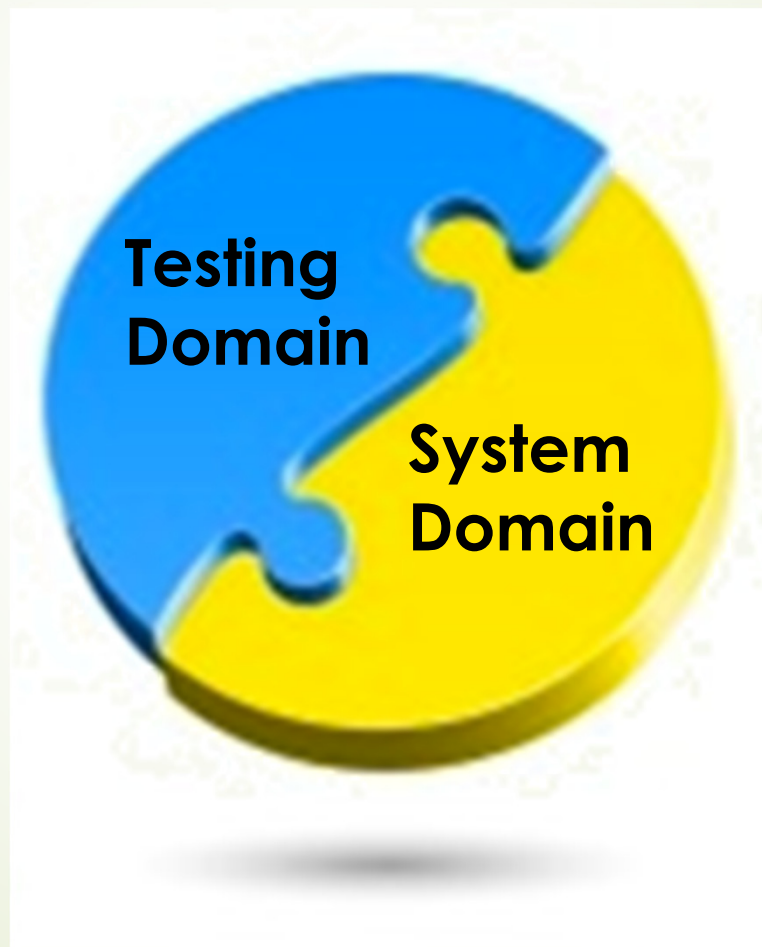
Can we utilise DSLs in Software Testing?

Test Case
Integration Test
Test Environment
Regression Test
Smoke Test
Accessibility Test
System Test
Test Suite
Performance Test



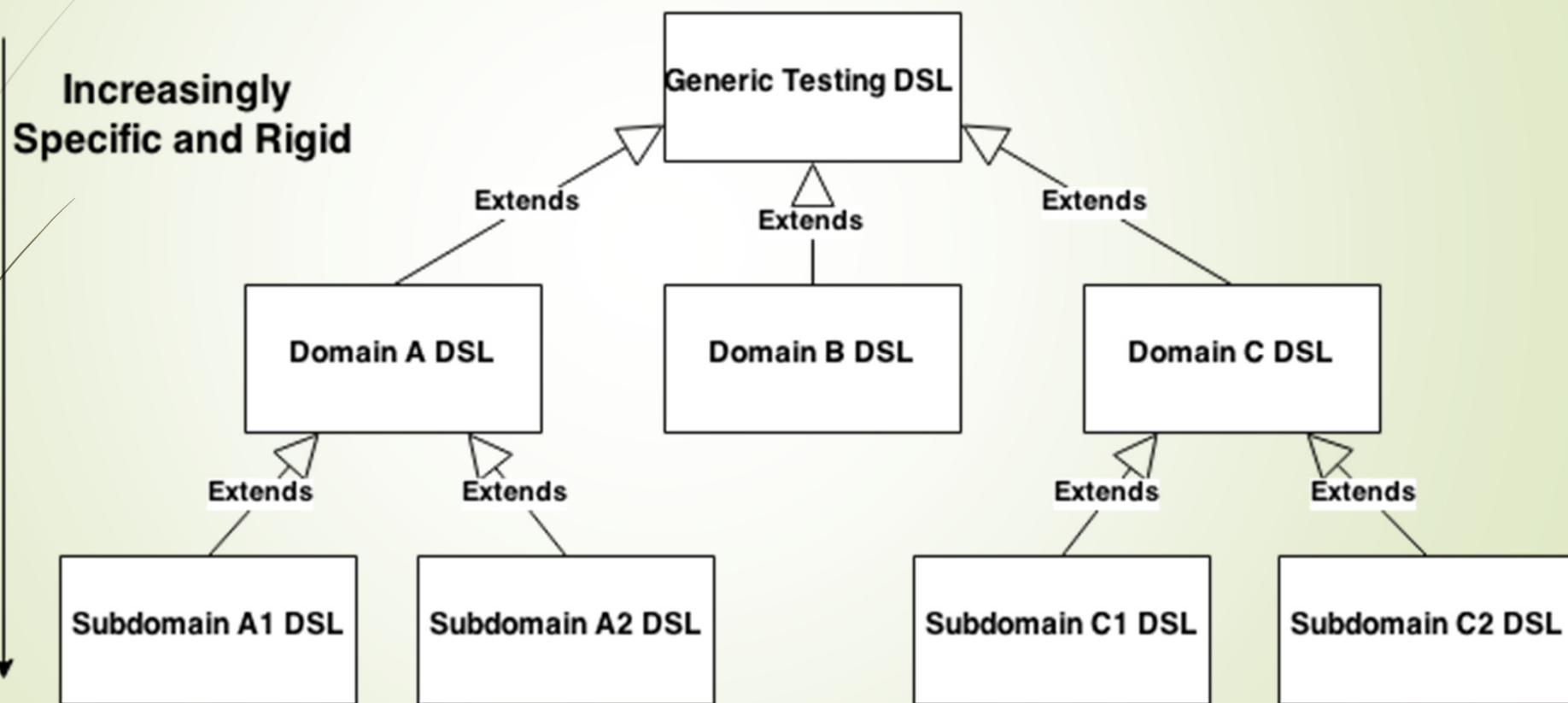


Can we utilise DSLs in Software Testing?





Can we utilise DSLs in Software Testing?





Vision

Tests expressed in an external DSL

Compiler/Interpreter

Automated Testing Framework

System Under Test



Challenges

- Language Design
- Engineering Challenge



Three Case Studies





Challenge 1: Language Design



Characteristics of a good DSL

1. Domain Specific
2. Simple
3. Reusable
4. General
5. Extensible
6. Similarity to other familiar languages
7. Complete
8. Orthogonal

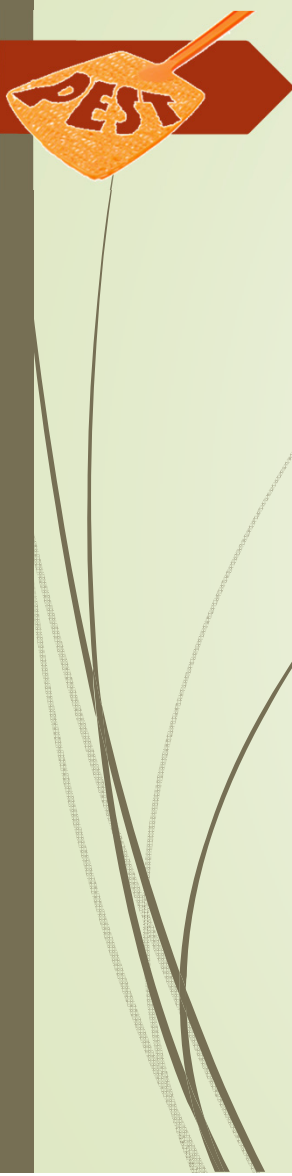




Common Language Elements

- Testing domain remains constant across case studies
- Typical notions:
 - Test Suites
 - Tests
 - Test life cycle
 - Test Setup
 - Exercising the SUT
 - Verification of expected behaviour
 - Test Tear Down
 - Reusable procedure-type mechanisms

Common Language Elements



```
DEFINE Test Suite "Calculator Test Suite"
```

```
    DEFINE Setup
```

```
        ...
```

```
    END
```

```
    DEFINE Test "Simple Addition"
```

```
        ...
```

```
    END
```

```
    DEFINE Test "Divide by zero"
```

```
        ...
```

```
    END
```

```
    DEFINE Teardown
```

```
        ...
```

```
    END
```

```
END
```



Engineering Challenges

- Interacting with the system under test
- Readability/Maintainability of generated code



Conclusions

- The use of DSLs for specifying and executing test has promise
- Different criteria of language design gain prominence depending on characteristics of games
- Engineering challenges are surmountable, especially if there is good cooperation with developers
- Questions
 - Who curates/owns the language?
 - What effect will a change on the language have on existing scripts?
 - How is that process controlled?
 - Who maintains code generators and how?



Future Work

- Gauge industry opinion
- Carry out more industry case studies
- Look at improving the current state-of-the-art in the industry