# Embedded DSL for System Testing via Scala

## Final Year Project - Final Evaluation

Rohit Mukherjee

April 15, 2015

Supervisor: Dr. Chin Wei Ngan
Project ID: H018570

# Table of Contents I

# Problem

## Lack of mature frameworks/DSLs for system testing

- **Dearth** of mature frameworks and tools available for system testing
- Existing Frameworks: **Not suitable** for system testing
- **Pressing** need for system level testing

# Existing Testing Frameworks


JUnit

ScalaTest™
simply productive™

# Types of System Testing

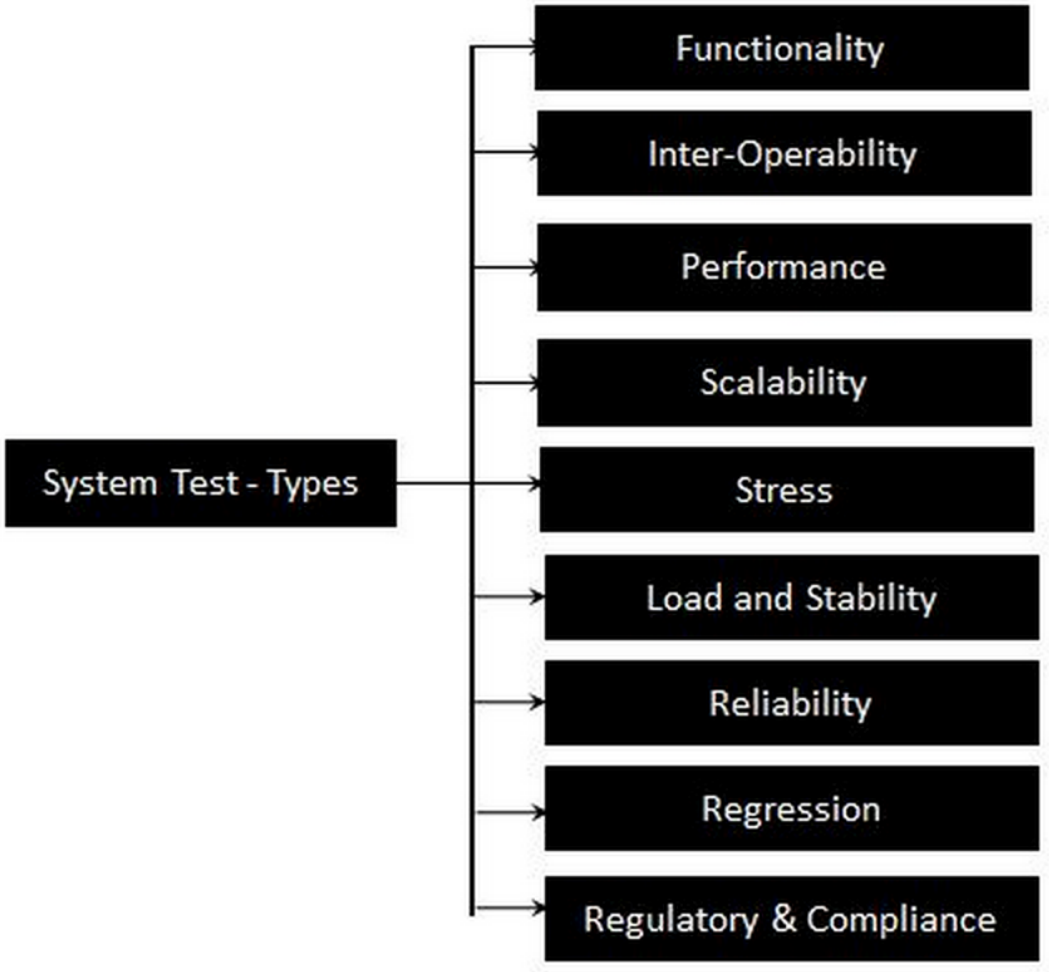



Figure: Different types of system testing

# Case Study for System Testing: Language Verification

Requirements:

- HIP Verification
- SLEEK Verification
- HIP/SLEEK Regression Testing
- HIP/SLEEK Verifier Performance Testing
- HIP/SLEEK Code Base Repository Testing
- Programming Language Prover Testing

# Existing System for HIP/SLEEK verification - Perl Script

```
     "mult_c", "SUCCESS",
     "shift_left", "SUCCESS",
     "mult", "SUCCESS",
     "karatsuba_mult", "SUCCESS",
     "is_zero", "SUCCESS",
     "is_equal", "SUCCESS",
     "compare2", "SUCCESS",
     "compare_int", "SUCCESS",
     "div_with_remainder", "SUCCESS"],
    ["append_imm.ss", 1,  " --imm ", "append", "SUCCESS"],
    ["kara.ss",1,  " --imm -tp redlog", "karatsuba_mult","SUCCESS"],
    ["kara-imm-star.ss",1,  " --imm -tp redlog" , "karatsuba_mult","SUCCESS"],
    ["kara-imm-conj.ss",1,  "--imm -tp redlog", "karatsuba_mult","SUCCESS"],
    ["ll_imm.ss", 6,  " --imm ", "length", "SUCCESS",
     "append", "SUCCESS",
     "sumN", "SUCCESS",
     "set_next", "SUCCESS",
     "get_next_next", "SUCCESS",
     "get_next", "SUCCESS"
    ]],
"imm-field" => [
    ["imspd.ss",2,"-tp oc --field-ann --etcsu1 ","check_pass","SUCCESS","login","SUCCESS"],
    ["getset.ss",5,"-tp oc --field-ann --etcsu1 ","sset","SUCCESS","get","SUCCESS","setA","SUCCESS","getA","S
    ["bigint.ss",15,"-tp redlog --field-ann --etcsu1 ","clone","SUCCESS","add_one_digit","SUCCESS","add_c","S
    ["insertion_simple.ss",1,"-tp oc --field-ann --etcsu1 ","insert","SUCCESS"],
    ["schorr-waite-list.ss",1,"-tp om --field-ann --etcsu1 ","lscan","SUCCESS"],
    ["sll.ss",4,"-tp oc --field-ann --etcsu1 ","delete","SUCCESS","get_tail","SUCCESS","insert","SUCCESS","in
],
```

Figure: Legacy Perl Script

# Existing System for HIP/SLEEK verification - Perl Script

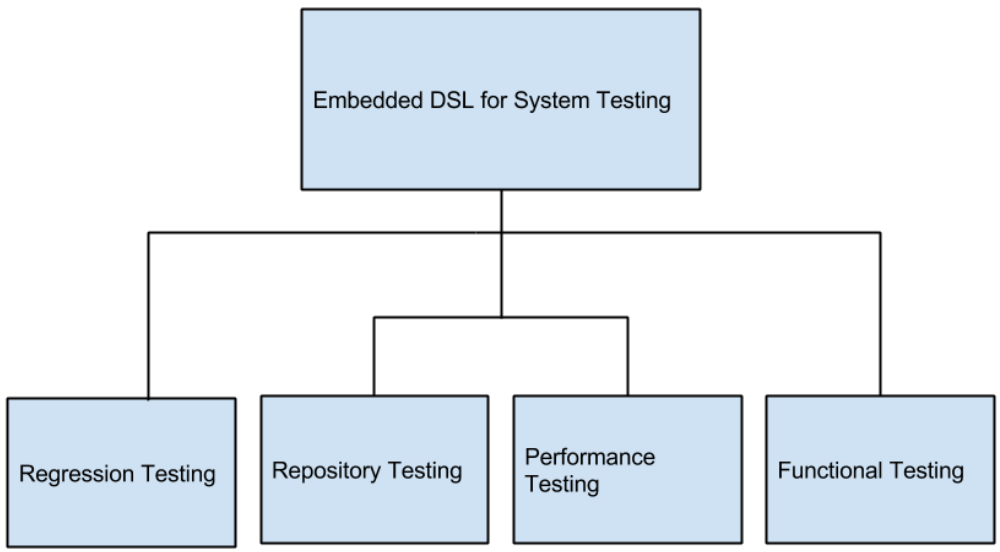
```
sub sleek_process_file  {
  foreach $param (@param_list)
  {
      #my $lem = -1; # assume the lemma checking is disabled by default; make $lem=1 if lemma che
      my $err = 0;
      my $barr = 0;
      if ("$param" =~ "musterr") {
          print "Starting sleek must/may errors tests:\n";
          $exempl_path_full = "$exec_path/errors";
          $err = 1;
      }
      if ("$param" =~ "sleek_barr"){ $barr=1;}

      if ("$param" =~ "sleek") {
          print "Starting sleek tests:\n";
          $exempl_path_full = "$exempl_path/sleek";
      }else {
          $exempl_path_full = "$exempl_path/sleek/$param";
          print "Starting sleek-$param tests:\n";
```

Figure: Legacy Perl Script

# Overview of Solution

The DSL written covers **Performance Testing, Regression Testing, Functional Testing and Repository Testing**.



# New System - DSL developed

```
    new RepositoryTest() called "sleek_tests"
    on "/home/rohit/hg/sleek_hip/" within 300
    storeOutputIn "/home/rohit/High-Performance-DSLs/Reporting Tool/"
    run "./sleek.sh" inDirectory "/home/rohit/High-Performance-DSLs/" build

firstTest runTests
```

Figure: Individual Test Case Creation

# New System - DSL developed

```
suite addTest ("hip", BASE_DIR + "term/e1.ss", " ", OUTPUT_DIR, "e1.out", " loop:  SUCCESS")
suite addTest ("hip", BASE_DIR + "term/ex1.ss", " ", OUTPUT_DIR, "ex1.out", " length:  SUCCESS,  app2:  SUCCESS")
suite addTest ("hip", BASE_DIR + "term/ex10.ss", " ", OUTPUT_DIR, "ex10.out", " loop:  SUCCESS")
suite addTest ("hip", BASE_DIR + "term/ex11.ss", " ", OUTPUT_DIR, "ex11.out", " bsearch:  SUCCESS")
suite addTest ("hip", BASE_DIR + "term/ex15.ss", " ", OUTPUT_DIR, "ex15.out", " loop:  SUCCESS,  f:  SUCCESS")
suite addTest ("hip", BASE_DIR + "term/ex16.ss", " ", OUTPUT_DIR, "ex16.out", " loop:  SUCCESS")
suite addTest ("hip", BASE_DIR + "term/ex2.ss", " ", OUTPUT_DIR, "ex2.out", " loop:  SUCCESS")
suite addTest ("hip", BASE_DIR + "term/ex3.ss", " ", OUTPUT_DIR, "ex3.out", " loop:  SUCCESS")
suite addTest ("hip", BASE_DIR + "term/ex4.ss", " ", OUTPUT_DIR, "ex4.out", " inc_loop:  SUCCESS")
suite addTest ("hip", BASE_DIR + "term/ex5.ss", " ", OUTPUT_DIR, "ex5.out", " foo:  SUCCESS")
suite addTest ("hip", BASE_DIR + "term/ex6.ss", " ", OUTPUT_DIR, "ex6.out", " Ack:  SUCCESS")
```

Figure: Test Suite Creation

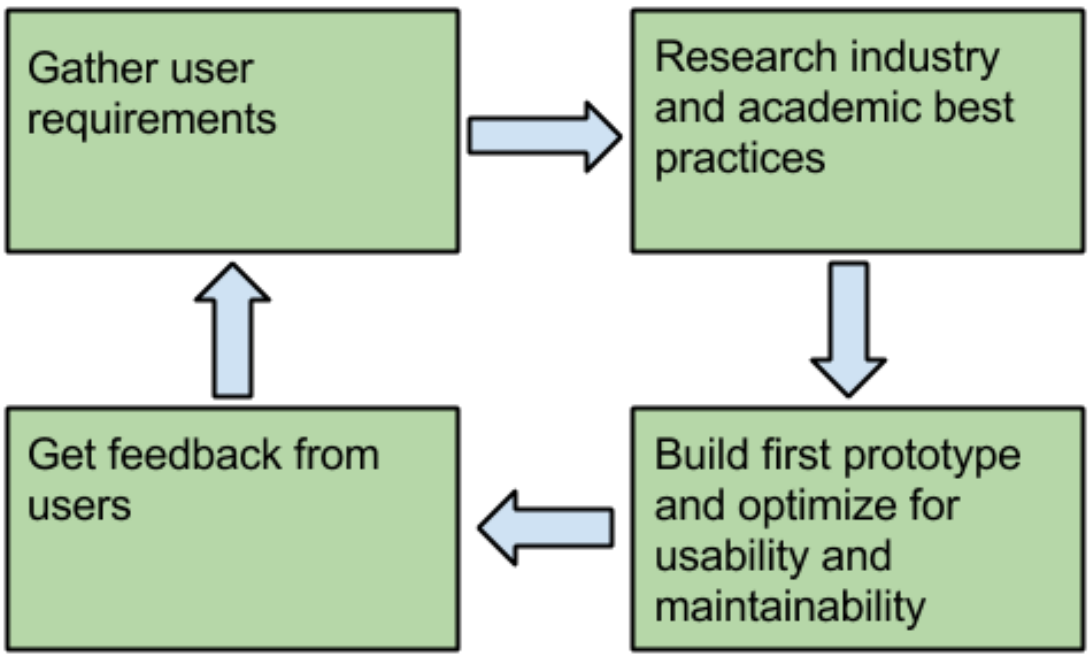
# Advantages of New System

- Extensibility
- Ensured Type Safety [17]
- Highly Configurable
- Lightweight Library
- Easy to use
- Domain Semantics [4]
- Integration with Version Controlled Work - flow

# Project Objectives

- Evaluate different DSL development techniques
- Choose most applicable technique
- Fulfill functional requirements of System Testing DSL

# Research Methodology



# Outline

# Microclassification of DSLs

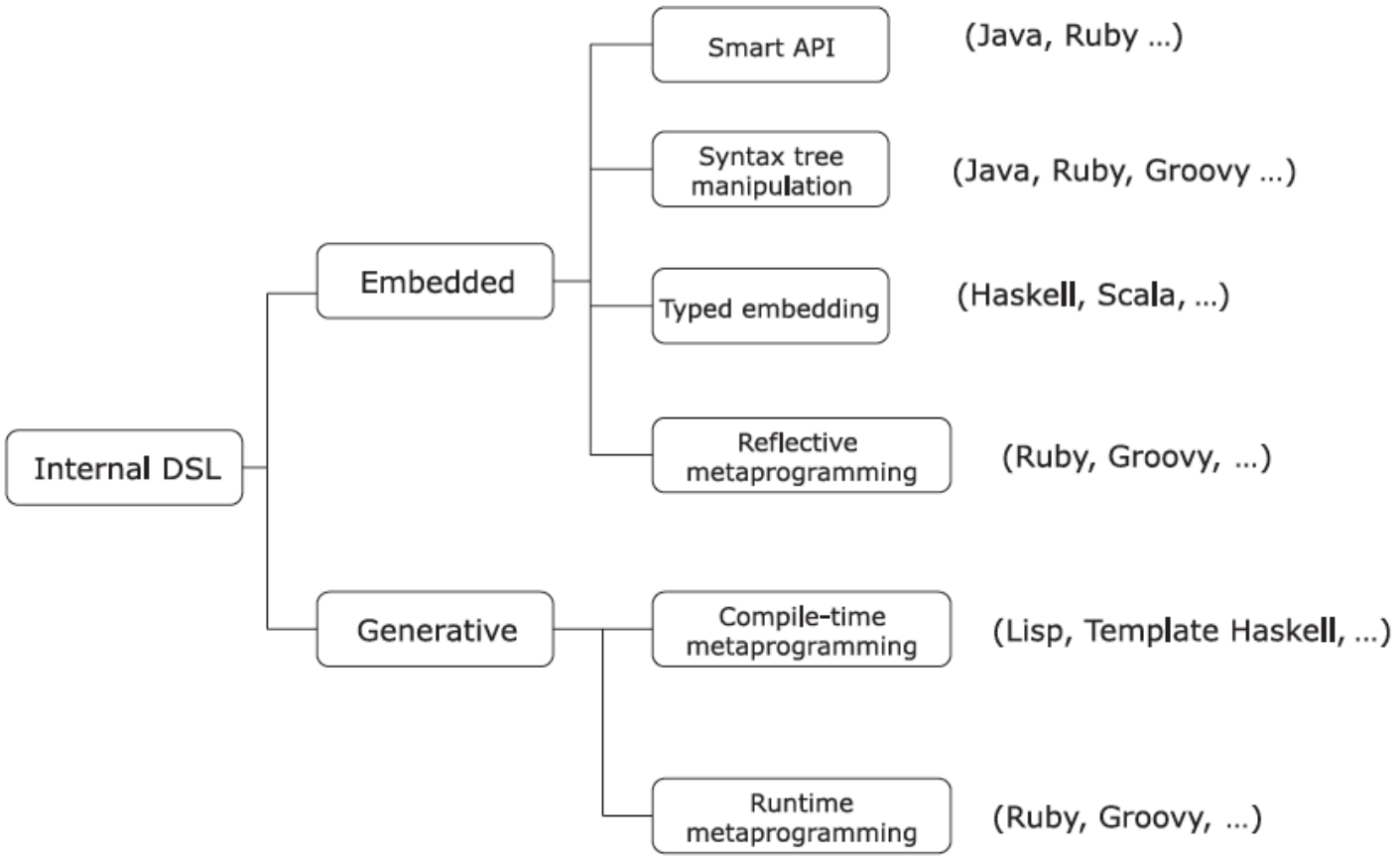



Figure: Classification of DSLs

# Choice of Embedded DSL approach

- Lightweight
- Compile Time Type Checking
- Easily maintainable and extensible
- Emphasis on Semantics
- Loose Coupling
- Representational Independence

```
1  new SleekTestCaseBuilder runCommand "sleek"
2  onFile "/home/rohit/hg/sleek_hip/examples/working/sleek/sleek9.slk"
3  withArguments "--elp" storeOutputInDirectory "results"
4  withOutputFileName "sleek9.out" checkAgainst "Valid, Fail, Valid, Valid"
```

Figure: Meaningful Semantics

# Choice of Programming Language

# Reasons for choosing Scala

| Feature | How Scala Does it |
|---|---|
| Flexible Syntax | 1. Optional dots in method invocation<br>2. Semicolon inference<br>3. Infix operators<br>4. Optional parentheses |
| Extensible Type System | 1. Scala shares Java's object model and extends it on many fronts<br>2. Traits for mix - in based inheritance<br>3. Case classes as value objects |
| Functional Support | 1. Scala supports both OO and Functional styles of programming<br>2. Functions in Scala are first class values and higher order functions are supported by the type system<br>3. Custom DSL control structures can be defined as closures and easily passed around<br>4. In pure OO, everything needs to be modelled as classes whereas in Scala, functional support allows closer domain modelling |

Figure: Motivation for using Scala

# Scala is a better choice

- Greater type safety due to compile time checking [4]
- Better collections hierarchy
- Expressiveness
- High readability
- Flexible Syntax
- Functional Programming Support

# Outline

# Architecture of DSL

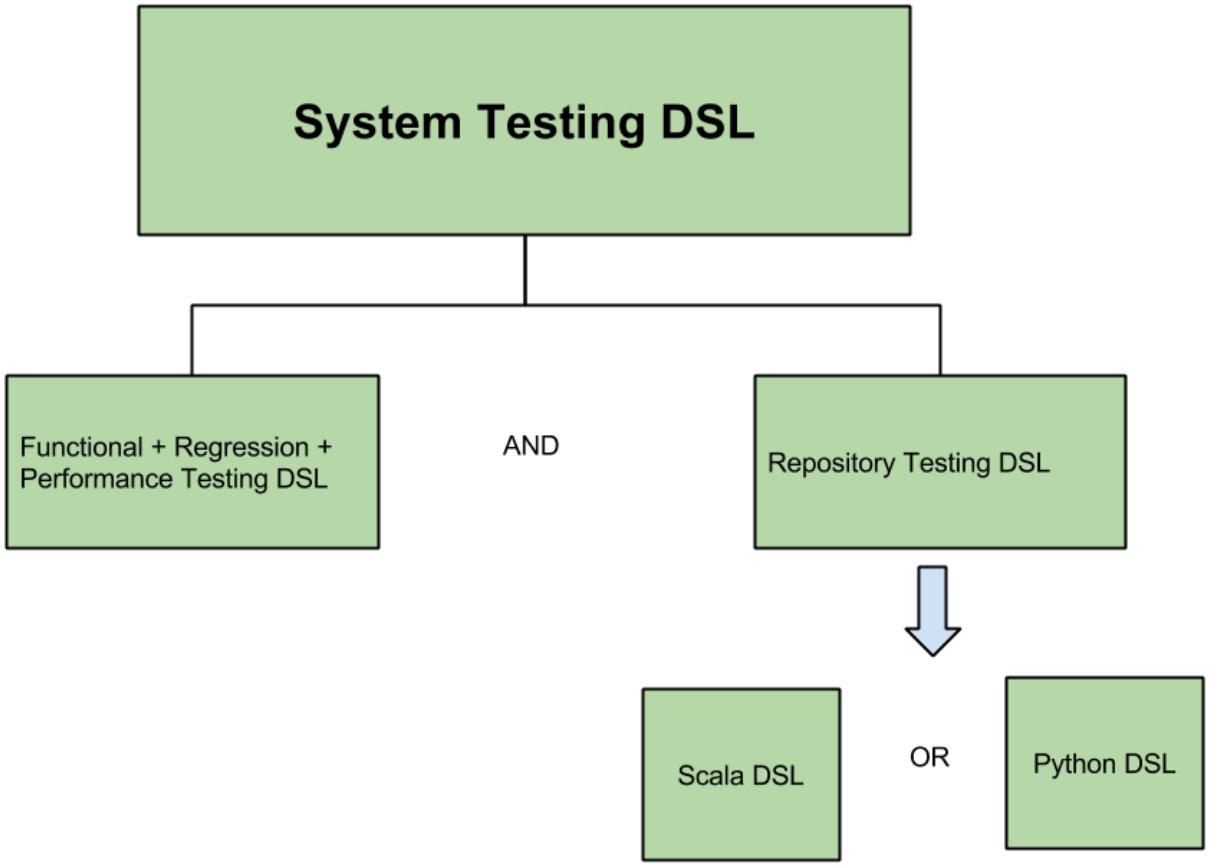



Figure: Architecture of System Testing DSL

# Composition of DSLs

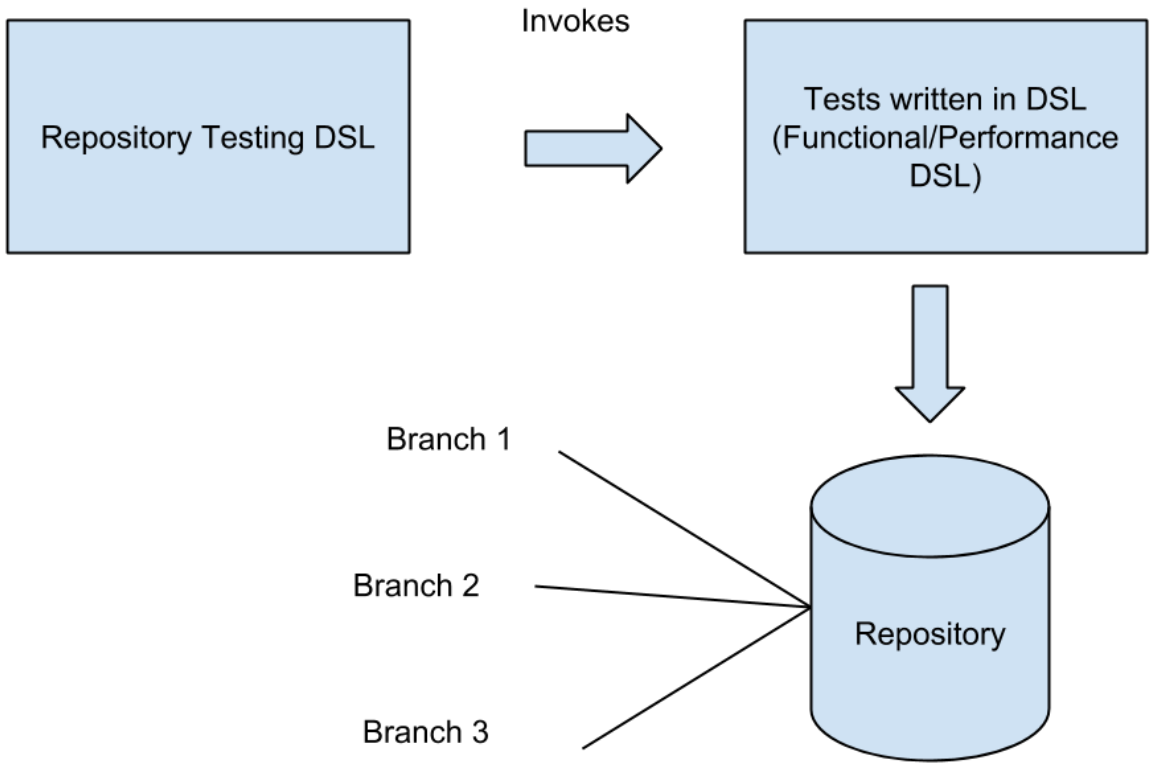



Figure: Composition Repository Testing and Functional/Performance Testing DSL

# Features

- Regression Testing
- Functional Testing
- Performance Testing
- Repository Testing

# Regression Testing

```
rohit@loris-82:~/High-Performance-DSLs$ sbt "run buildReference"
[info] Loading project definition from /home/rohit/High-Performance-DSLs/project
[info] Set current project to SystemTestingDSL (in build file:/home/rohit/High-Performance-DSLs/)
[info] Formatting 1 Scala source {file:/home/rohit/High-Performance-DSLs/}high-performance-dsls(compile) ...
[info] Compiling 1 Scala source to /home/rohit/High-Performance-DSLs/target/scala-2.10/classes...
[info] Running systemTestingDSL.Main buildReference
******************

Building References

******************

hip --print-min /home/rohit/hg/sleek_hip/baga/t/under2.ss
hip --print-min /home/rohit/hg/sleek_hip/baga/t/heaps.ss
hip --print-min /home/rohit/hg/sleek_hip/baga/t/ll-3.ss
hip --print-min /home/rohit/hg/sleek_hip/baga/t/under.ss
hip --print-min /home/rohit/hg/sleek_hip/baga/t/cll-d.ss
hip --print-min /home/rohit/hg/sleek_hip/baga/t/ll-3b.ss
hip --print-min /home/rohit/hg/sleek_hip/baga/t/ll-3a.ss
hip --print-min /home/rohit/hg/sleek_hip/baga/t/dll.ss
hip --print-min /home/rohit/hg/sleek_hip/baga/t/pr.ss
hip --print-min /home/rohit/hg/sleek_hip/baga/t/dll-td2.ss
hip --print-min /home/rohit/hg/sleek_hip/baga/t/ll-3d.ss
hip --print-min /home/rohit/hg/sleek_hip/baga/t/modular.ss
hip --print-min /home/rohit/hg/sleek_hip/baga/t/ll-3f.ss
hip --print-min /home/rohit/hg/sleek_hip/baga/t/ll-3c.ss
hip --print-min /home/rohit/hg/sleek_hip/baga/t/cll.ss
hip --print-min /home/rohit/hg/sleek_hip/baga/t/dll-modular.ss
hip --print-min /home/rohit/hg/sleek_hip/baga/t/dll-append-mod.ss
```

Figure: Building References for Regression Testing

# Regression Testing

## Running Regression Tests

3 primary options provided - **buildReference, runReference and overrideReference**. The first one creates a repository of reference results for specified tests. The second option runs a set of tests against the stored references. The last option is for selectively rebuilding reference results.

# Functional Testing - Sleek

```
sleek   /home/rohit/hg/sleek_hip/examples/working/sleek/infer/infer12.slk
Failed

Had: Entail (3) : Valid.
Expected:  Fail


sleek   /home/rohit/hg/sleek_hip/examples/working/sleek/infer/infer12.slk
Passed


sleek  --sa-en-cont /home/rohit/hg/sleek_hip/examples/working/sleek/infer/infer13.slk
Failed

Had: Entail (5) : Fail.(may) cause:UnionR[AndR[ ((x=1 & y=2 & flted_37_152!=y) | (x=1 & y=flted_37_152)) |-  x=z. LOCS:[0;11];  ((x=1 & y=
2)) |-  flted_37_152=null. LOCS:[0;11;37] (may-bug).],AndR[ (((x=1 & y=2 & z=3 & flted_37_164!=y & inf_p_141!=z) | (x=1 & y=2 &
Expected:  Valid


sleek  --sa-en-pure-field /home/rohit/hg/sleek_hip/examples/working/sleek/infer/infer14.slk
Passed


sleek   /home/rohit/hg/sleek_hip/examples/working/sleek/infer/infer15.slk
Failed

Binary failed to execute. Please investigate


sleek   /home/rohit/hg/sleek_hip/examples/working/sleek/infer/infer16.slk
Failed

Had: Entail (5) : Fail.(may) cause:AndR[ R(fld5) |-  fld5=null. LOCS:[37;39];  inf_p_40=null & inf_p_40!=x & x!=null |-  inf_p_40=z. LOCS:
[38;39] (may-bug).]
Expected:  Valid


sleek   --imm --en-imm-inv --etcsu1  /home/rohit/hg/sleek_hip/examples/working/sleek/ann2.slk
Passed
```

Figure: Running Sleek Tests Console Output

# Functional Testing - Sleek

## Sleek Tests

sleek /home/rohit/hg/sleek_hip/examples/working/sleek/sleek.slk

Passed

sleek /home/rohit/hg/sleek_hip/examples/working/sleek/cll-d.slk

Passed

sleek --dis-eps /home/rohit/hg/sleek_hip/examples/working/sleek/label-basic.slk

Passed

sleek --dis-eps /home/rohit/hg/sleek_hip/examples/working/sleek/label-dll.slk

Failed

()() sleek /home/rohit/hg/sleek_hip/examples/working/sleek/sleek1.slk

Passed

sleek /home/rohit/hg/sleek_hip/examples/working/sleek/sleek10.slk

Passed

Figure: Running Sleek Tests HTML Output

# Performance Testing

```
rohit@loris-82:~/High-Performance-DSLs/results$ cat sleek_performance_report_06_55_15.perf
/home/rohit/hg/sleek_hip/examples/working/sleek/infer/infer5.slk
Runtime was 1071 milliseconds
/home/rohit/hg/sleek_hip/examples/working/sleek/infer/infer5a.slk
Runtime was 1018 milliseconds
/home/rohit/hg/sleek_hip/examples/working/sleek/lemmas/lseg_case.slk
Runtime was 1391 milliseconds
/home/rohit/hg/sleek_hip/examples/working/sleek/lemmas/ll_tail.slk
Runtime was 1566 milliseconds
/home/rohit/hg/sleek_hip/examples/working/sleek/fracperm/split_simple.slk
Runtime was 3541 milliseconds
/home/rohit/hg/sleek_hip/examples/working/sleek/veribsync/barrier-dynamic2.slk
Runtime was 3236 milliseconds
/home/rohit/hg/sleek_hip/examples/working/sleek/../tree_shares/barrier.slk
Runtime was 12611 milliseconds
/home/rohit/hg/sleek_hip/examples/working/sleek/../tree_shares/barrier3.slk
Runtime was 10862 milliseconds
/home/rohit/hg/sleek_hip/examples/working/sleek/../tree_shares/fractions.slk
Runtime was 1175 milliseconds
/home/rohit/hg/sleek_hip/examples/working/sleek/fracperm/sleek8.slk
Runtime was 7789 milliseconds
/home/rohit/hg/sleek_hip/examples/working/sleek/fracperm/sleek9.slk
Runtime was 1627 milliseconds
/home/rohit/hg/sleek_hip/examples/working/sleek/fracperm/norm3.slk
Runtime was 1858 milliseconds
/home/rohit/hg/sleek_hip/examples/working/sleek/fracperm/split_simple.slk
Runtime was 3533 milliseconds
```

Figure: Performance Report for Sleek Tests

# Repository Testing

```
(Running test on commit ,26dbee0d4dd37ad004d5f81477d85d8ec2c8663a)
(./sleek.sh,/home/rohit/High-Performance-DSLs/)
Output stored in directory /home/rohit/High-Performance-DSLs/Reporting Tool//ti_exp/sleek_tests/sleek_tests14482_Wed Mar 18 (
(Running test on commit ,fd155de57e8cc9a288df923864d72cdb1d6804fa)
(./sleek.sh,/home/rohit/High-Performance-DSLs/)
Output stored in directory /home/rohit/High-Performance-DSLs/Reporting Tool//ti_exp/sleek_tests/sleek_tests14481_Tue Mar 17 2
(Running test on commit ,4ebc77db061d89b74a31fb8493a7fd751aef921b)
(./sleek.sh,/home/rohit/High-Performance-DSLs/)
Output stored in directory /home/rohit/High-Performance-DSLs/Reporting Tool//ti_exp/sleek_tests/sleek_tests14480_Tue Mar 17 2
(Running test on commit ,117991577fd13a6be01c46b727fc0e70735c5d95)
(./sleek.sh,/home/rohit/High-Performance-DSLs/)
Output stored in directory /home/rohit/High-Performance-DSLs/Reporting Tool//ti_exp/sleek_tests/sleek_tests14479_Tue Mar 17 2
(Running test on commit ,d47b99af55f2f1aca42edde8e4eed0e3337217c0)
(./sleek.sh,/home/rohit/High-Performance-DSLs/)
Output stored in directory /home/rohit/High-Performance-DSLs/Reporting Tool//ti_exp/sleek_tests/sleek_tests14478_Tue Mar 17 2
(Running test on commit ,3ca560da370335730adc2e1637eaa14da247adb7)
(./sleek.sh,/home/rohit/High-Performance-DSLs/)
Output stored in directory /home/rohit/High-Performance-DSLs/Reporting Tool//ti_exp/sleek_tests/sleek_tests14477_Tue Mar 17 2
(Running test on commit ,00a5f5299ee77f22de1e8c404c3ec2b49e9bd84a)
(./sleek.sh,/home/rohit/High-Performance-DSLs/)
Output stored in directory /home/rohit/High-Performance-DSLs/Reporting Tool//ti_exp/sleek_tests/sleek_tests14476_Tue Mar 17 2
(Running test on commit ,1f62e178a5461fa0bffb569de7e19e166b648985)
(./sleek.sh,/home/rohit/High-Performance-DSLs/)
Output stored in directory /home/rohit/High-Performance-DSLs/Reporting Tool//ti_exp/sleek_tests/sleek_tests14475_Sun Mar 08 2
(Running test on commit ,088e1a57b3234292d0772a15593ed42ad124f1a5)
(./sleek.sh,/home/rohit/High-Performance-DSLs/)
Output stored in directory /home/rohit/High-Performance-DSLs/Reporting Tool//ti_exp/sleek_tests/sleek_tests14474_Tue Mar 17 2
(Running test on commit ,486de723f2bd634cc3b6427f9ec38f2d97a8b31f)
(./sleek.sh,/home/rohit/High-Performance-DSLs/)
Output stored in directory /home/rohit/High-Performance-DSLs/Reporting Tool//ti_exp/sleek_tests/sleek_tests14473_Tue Mar 17 1
(Running test on commit ,eb24fc444399261e5a049dda24ba714a7a22113a)
```

Figure: Repository level tests being run on the HIP/SLEEK code base

# Test Statistics

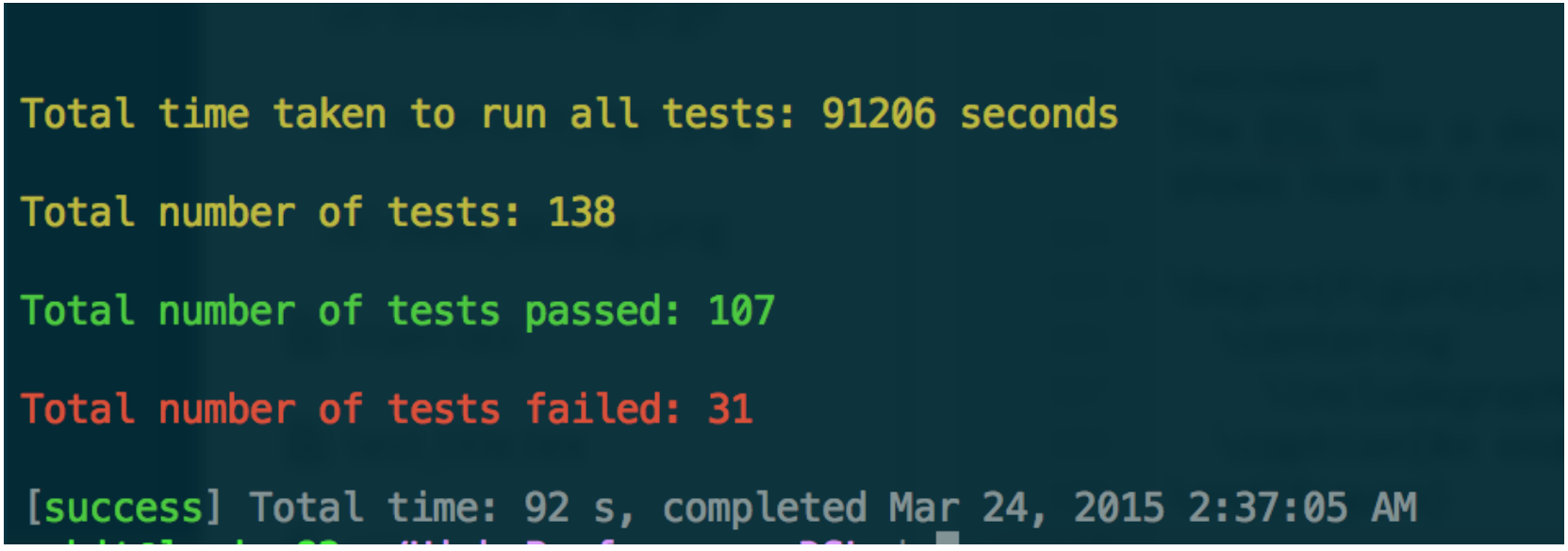



Figure: Test statistics at the end of every test Console Output

# Test Statistics

sleek --en-para -perm bperm -tp redlog /home/rohit/hg/sleek_hip/examples/working/sleek/veribsync/barrier

Passed

sleek --en-para -perm bperm -tp redlog /home/rohit/hg/sleek_hip/examples/working/sleek/veribsync/barrier

Passed

sleek --en-para -perm bperm -tp redlog /home/rohit/hg/sleek_hip/examples/working/sleek/veribsync/barrier

Passed

Total number of tests: 138

Total number of tests passed: 101

Total number of tests failed: 37

[success] Total time: 127 s, completed Feb 12, 2015 4:53:58 AM

Figure: Test statistics at the end of every test HTML Output

# Test Statistics

sleek --en-para -perm bperm -tp redlog /home/rohit/hg/sleek_hip/examples/working/sleek/veribsync/barrier

Passed

sleek --en-para -perm bperm -tp redlog /home/rohit/hg/sleek_hip/examples/working/sleek/veribsync/barrier

Passed

sleek --en-para -perm bperm -tp redlog /home/rohit/hg/sleek_hip/examples/working/sleek/veribsync/barrier

Passed

Total number of tests: 138

Total number of tests passed: 101

Total number of tests failed: 37

[success] Total time: 127 s, completed Feb 12, 2015 4:53:58 AM

Figure: Test statistics at the end of every test HTML Output

Demo

# Outline

4 Conclusion
- Summary of Work
- Innovation
- Future Work
- Key Takeaways

# Summary of Work and Goals Achieved

- An innovative way of performing system level tests using functional, regression, repository and performance testing techniques
- An extensible DSL for functional, performance and regression testing on a system level
- Migration of a 2500 line **Perl Script** to DSL
- Several applications of the system were found - **HIP Verification, SLEEK verification, Repository and Regression Testing**
- A DSL was developed in Python and Scala which can test all branches/commits of a mercurial project and summarize the results in an organized manner

# Innovation

## Innovation

There are several unit testing and web testing frameworks present in the software testing ecosystem today. However, there are very few tools that perform system level tests. The DSLs implemented fill in this gap in the testing ecosystem by providing flexible ways of performing regression, functional and performance testing. The **Repository Testing** feature provided is an innovative and effective technique to test the source code repository when several developers are contributing to it.

# Limitations and Future Work

- Improvement of Performance Testing features including graphical user interfaces
- Cleaner syntax that abstracts away more details of host language, Scala
- Run - time performance optimization through some meta - programming
- Integrate **Git, SVN** or other version control tools with the reporting tool in Python
- Extension of DSL to make it more generic and applicable to different applications for system testing

# Key Takeaways

- Importance of system testing
- Insight into DSL development and **industry best practices**
- Exposure to the functional and OOP idioms and constructs in **Scala** and **Python**
- Understanding of how testing frameworks (JUnit/ScalaTest) are built

# Project Statistics

- 80% of the project was written in **Scala**
- The total number of lines of code in the system are approximately **10,000**
- The project has **228 commits** and 7 branches

A graph showing commit frequency between September and April is shown below:

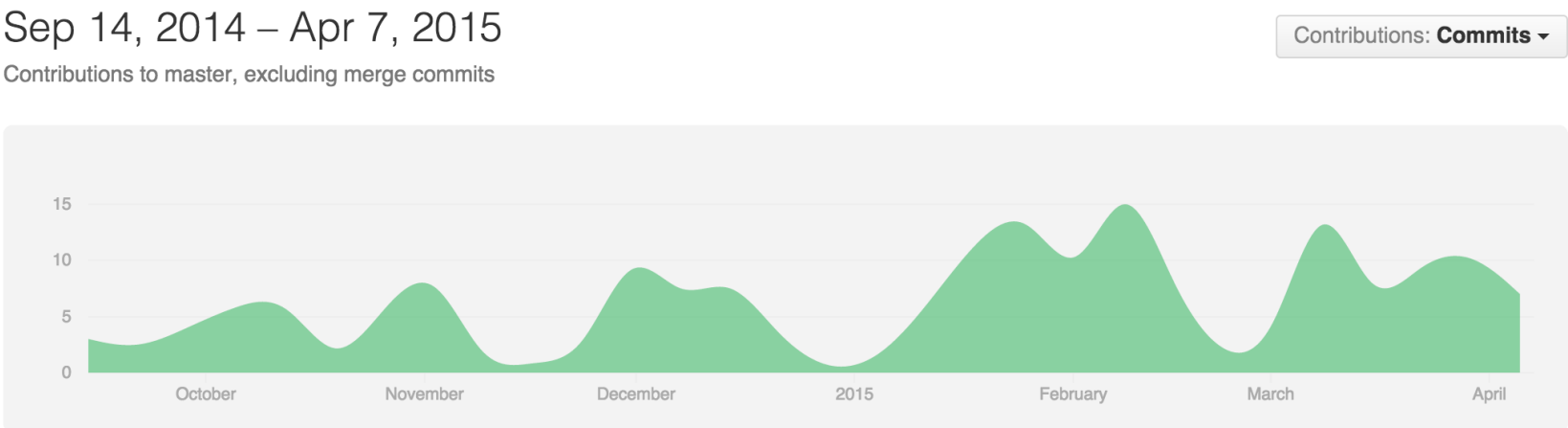



Figure: Commit frequency statistics generated by Github

# Q&A

Thank You!

# Appendix 1 - UML for Repository Testing DSL

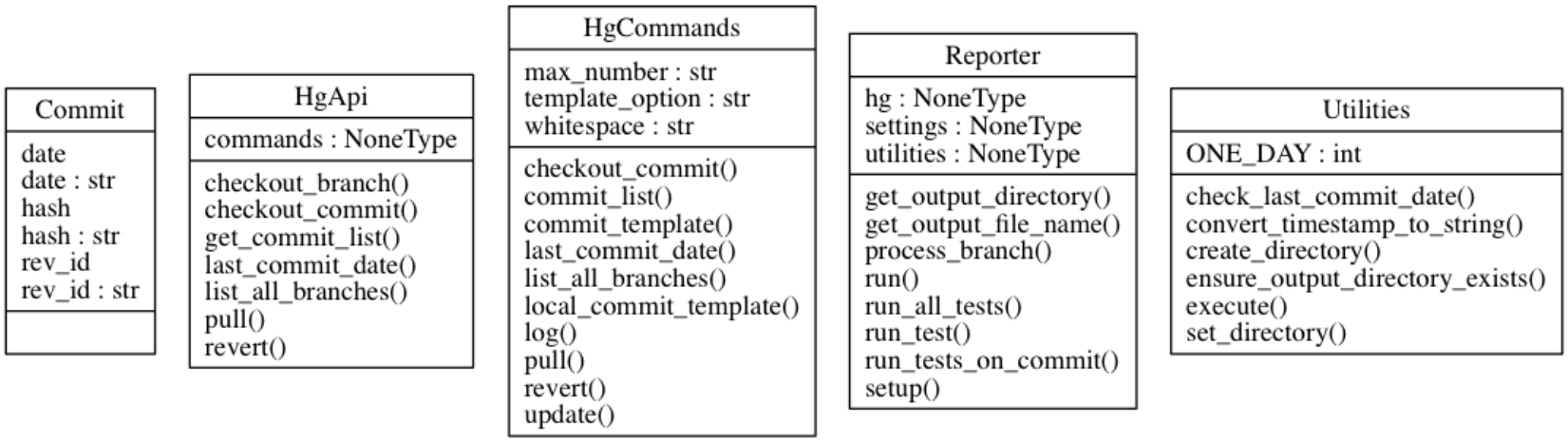



Figure: UML Diagram for Repository Testing DSL

# Appendix 2 - Design Patterns

- Singleton
- Factory Pattern
- Builder Pattern
- Future Await Pattern

# Appendix 3 - YAML for configuration

**YAML** was chosen as the markup language for configuration language for the DSLs. An example of the YAML markup is shown below:

```yaml
# Settings File
---

# Directory settings for machines
repository:
  remote:
  loris_local: /home/rohit/hg/sleek_hip/
  local: /Users/rohitmukherjee/dev/repositories/scalaWorkspace/High-Performance-DSLs/

# Application Settings
app:
  # Time period to run tests in days
  time_period: 180  # Run tests for commits made in the last months
  output_directory_name: Sleek_Test_Results
  # Must end with a '/'
  output_directory_location: /home/rohit/High-Performance-DSLs/Reporting Tool/
```

Figure: settings.yaml - used to configure the reporting tool

# Appendix 4 - Reasons for choosing YAML

- Human Readability (more so than XML/JSON)
- Low number of format characters
- Language agnostic, can be parsed easily by different languages
- Sufficient types of data structures
- Not verbose like XML
- Faster than XML
- Python - like syntax

# Appendix 5 - Reasons for choosing sbt as build tool

- Incremental compilation
- Interactive Shell
- Native support for compiling Scala code and integrating with many Scala test frameworks
- Build descriptions written in a Scala DSL[15]
- Dependency Resolution

# References



Chin, W. (2011, January 1). http:cristian.pl-bits.ro/Publications/OOPSLA-Comp_2011_demo.pdf. Retrieved March 31, 2015, from http:cristian.pl-bits.ro/Publications/OOPSLA-Comp_2011_demo.pdf


Dingsyr, T., Nerur, S., Balijepally, V. and Moe, N. (2012). A decade of agile methodologies: Towards explaining agile software development. Journal of Systems and Software, 85(6), pp.1213-1221.


Fowler, M. (2005, 12 20). Fluent interface. Retrieved from http://martinfowler.com/bliki/FluentInterface.html


Ghosh, D. (2011). DSLs in Action (1st ed., pp. 20 - 87). Connecticut: Manning.

# References



Hetzel, William C., The Complete Guide to Software Testing, 2nd ed. Publication info: Wellesley, Mass. : QED Information Sciences, 1988. ISBN: 0894352423.Physical description: ix, 280 p. : ill ; 24 cm.


Hundt, R. (2011, June 6). Loop Recognition in C /Java/Go/Scala. Retrieved March 30, 2015.


Introduction to Test Driven Development (TDD). (n.d.). Retrieved November 4, 2014.


Gamma, E. (1995). Design patterns: Elements of reusable object-oriented software. Reading, Mass.: Addison-Wesley.


Lejdfors/Lund University, C. (2006). Techniques for implementing embedded domain specific languages in dynamic languages (Doctoral dissertation, Lund University, Lund, Sweden).

# References



Odersky, M., & Rompf, T. (n.d.). Lightweight Modular Staging: A Pragmatic Approach to Runtime ode Generation and Compiled DSLs. *GPCE 2010, October 1013, 2010*. Retrieved from http://infoscience.epfl.ch/record/150347/files/gpce63-rompf.pdf


Odersky, M., & Sujeeth, A. (2013). Forge: Generating a High Performance DSL Implementation from a Declarative Specication. *GPCE 2013: 12th International Conference on Generative Programming: Concepts & Experiences*. Retrieved November 4, 2014, from http://ppl.stanford.edu/papers/gpce13-sujeeth.pdf


PEP 8 - Style Guide for Python Code — Python.org. (n.d.). Retrieved from https://www.python.org/dev/peps/pep-0008/


Picking an Interpreter  The Hitchhiker's Guide to Python. (n.d.). Retrieved from http://docs.python-guide.org/en/latest/starting/which-python/

# References



Ragnarsson/Reykjavik University, . A. (n.d.). Importance of Design Patterns and Frameworks for Software Development (Unpublished doctoral dissertation). Reykjavk University, Reykjavk, Iceland.


The interactive build tool (sbt) http://www.scala-sbt.org/


The Python Programming Language. (n.d.). Retrieved March 31, 2015, from https://www.python.org/doc/


The Scala Programming Language (The Scala Programming Language) http://www.scala-lang.org/

# References



The Official YAML Web Site. (n.d.). Retrieved from http://yaml.org/


Twitter. (n.d.). Scala School. Retrieved from https://twitter.github.io/scala_school/


Williams, L., Kudrjavets, G., & Nagappan, N. (2008). On the Effectiveness of Unit Test Automation at Microsoft.


YAML Tutorial - Xavier Shay's Blog. (n.d.). Retrieved from http://rhnh.net/2011/01/31/yaml-tutorial