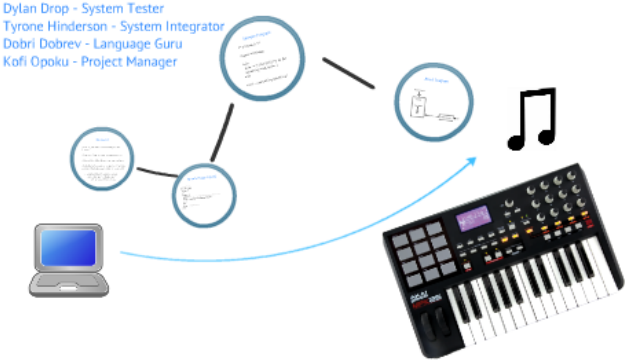
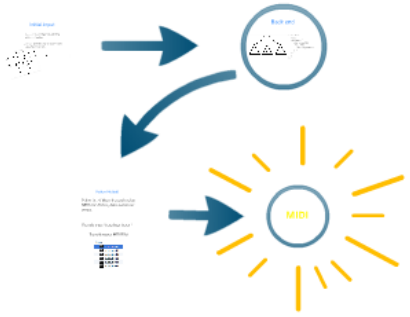
SCALR
The MIDI Programming Language

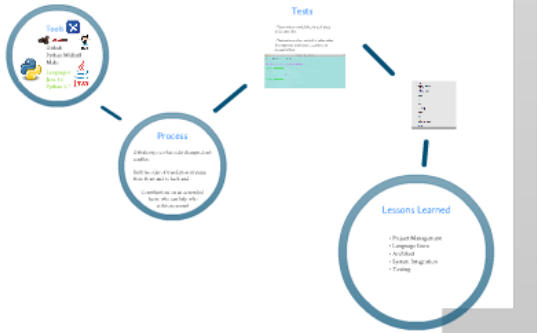
Mark Aligbe - System Architect
Dylan Drop - System Tester
Tyrone Hinderson - System Integrator
Dobri Dobrev - Language Guru
Kofi Opoku - Project Manager

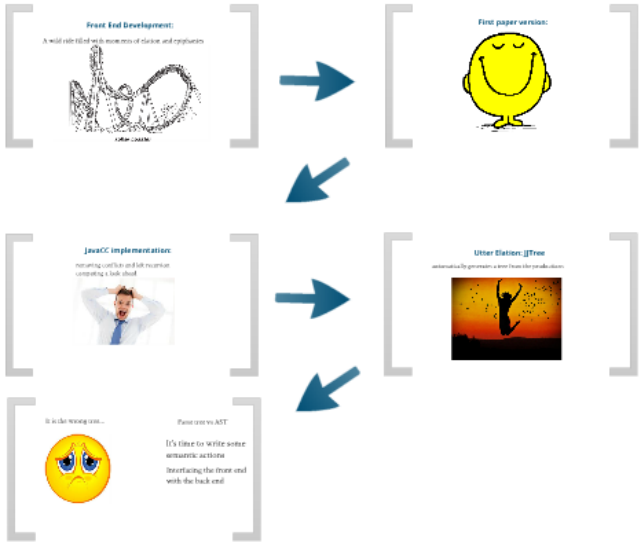


Steps to Transform SCALR into MIDI



Tools, Processes, Conclusions

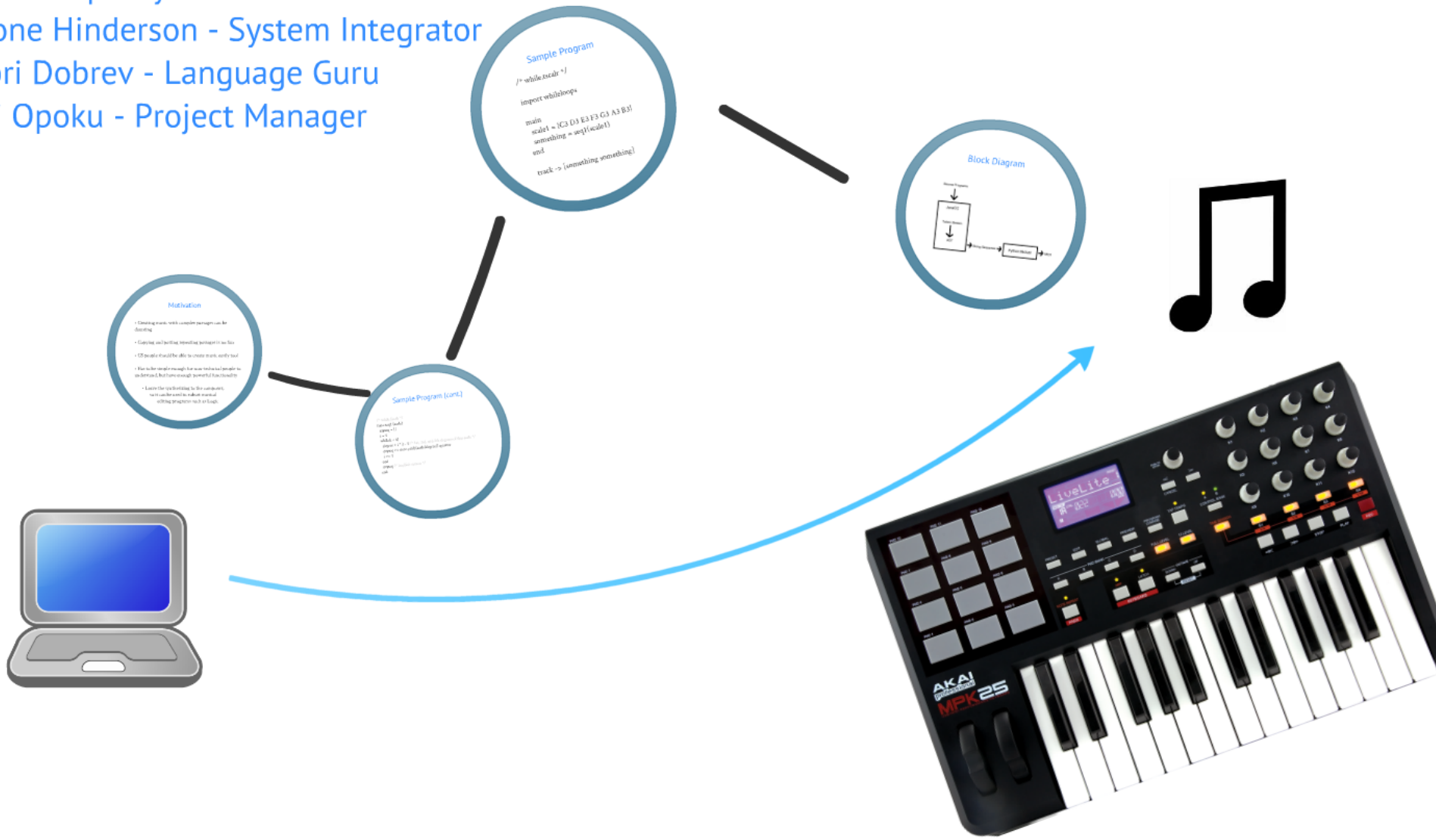




SCALR

The MIDI Programming Language

Mark Aligbe - System Architect
Dylan Drop - System Tester
Tyrone Hinderson - System Integrator
Dobri Dobrev - Language Guru
Kofi Opoku - Project Manager



Motivation

- Creating music with complex passages can be daunting

- Copying and pasting repeating passages is no fun

- CS people should be able to create music easily too!

- Has to be simple enough for non-technical people to understand, but have enough powerful functionality

- Leave the synthesizing to the composer, so it can be used in robust musical editing programs such as Logic

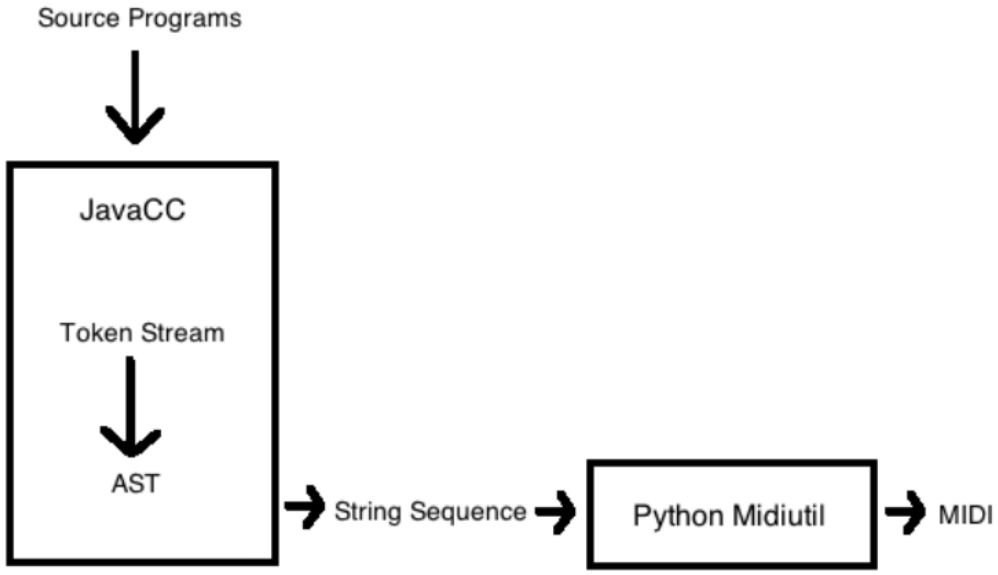
Sample Program (cont.)

```
/* while.fscalr */
func seq1 (scale)
 myseq = []
 i = 1
 while(i < 4)
  degree = i * 2 - 1 /* 1st, 3rd, and 5th degrees of this scale */
  myseq += note.pitch(scale[degree]).quarter
  i += 1
 end
 myseq /* implicit return */
end
```

Sample Program

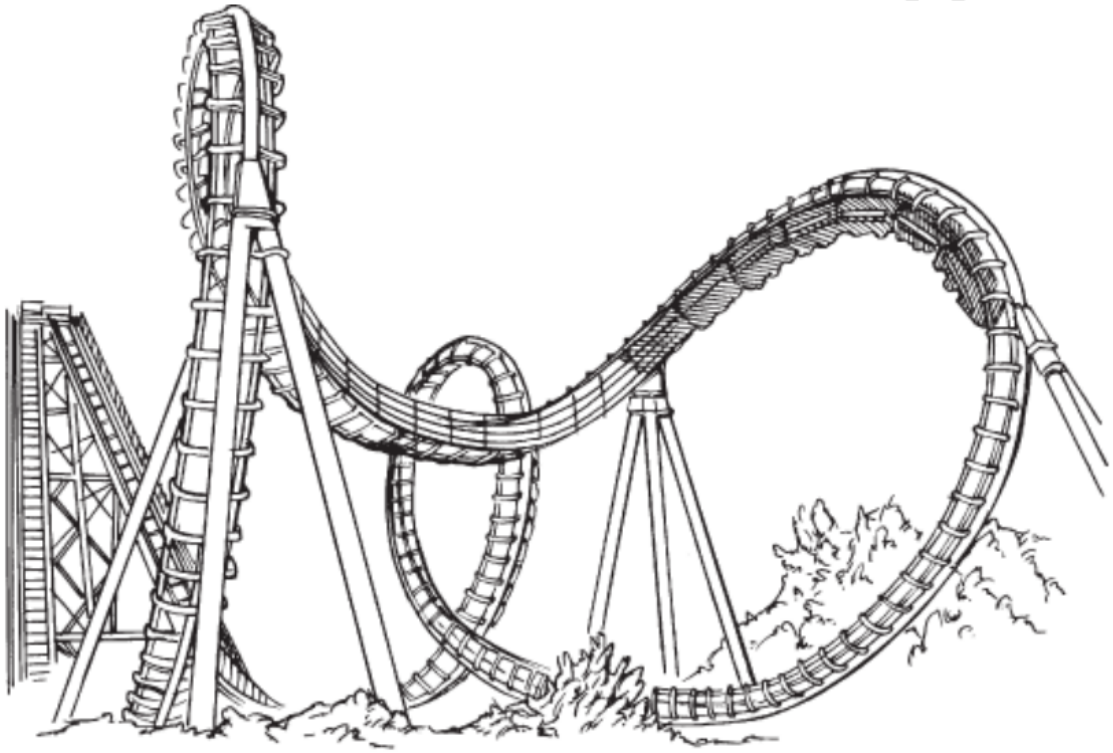
```
/* while.tscalr */

import whileloops

main
  scale1 = {C3 D3 E3 F3 G3 A3 B3}
  something = seq1(scale1)
end

track -> [something something]
```

Block Diagram



Front End Development:

A wild ride filled with moments of elation and epiphanies



roller coaster

First paper version:

JavaCC implementation:

removing conflicts and left recursion

computing a look ahead

Utter Elation: JJTree

automatically generates a tree from the productions

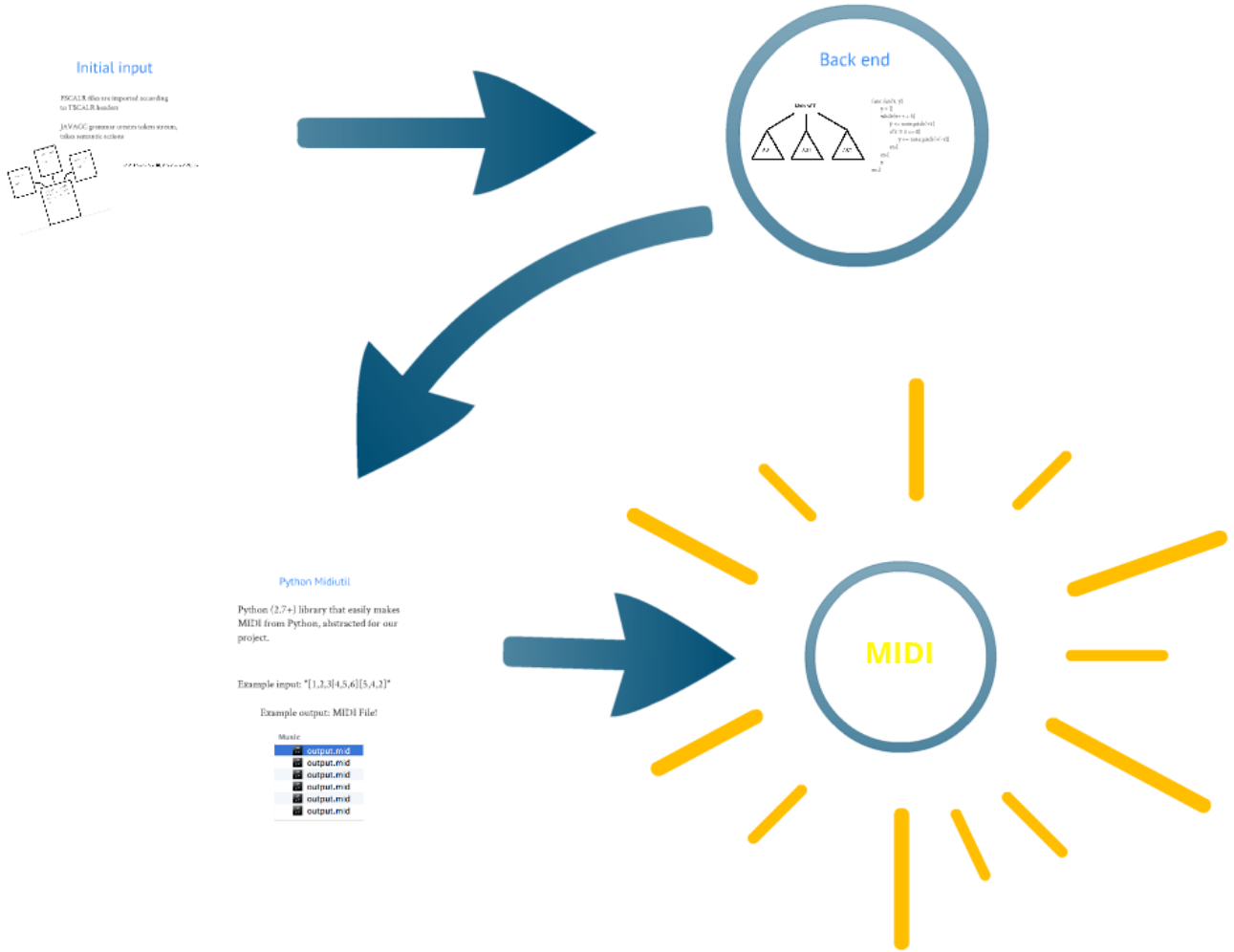
It is the wrong tree...

Parse tree vs AST

It's time to write some semantic actions

Interfacing the front end with the back end

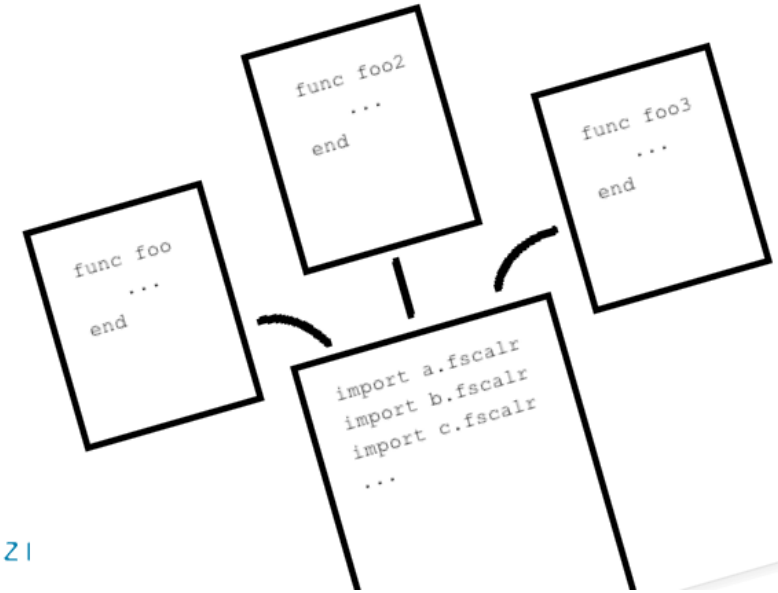
Steps to Transform SCALR into MIDI



Initial input

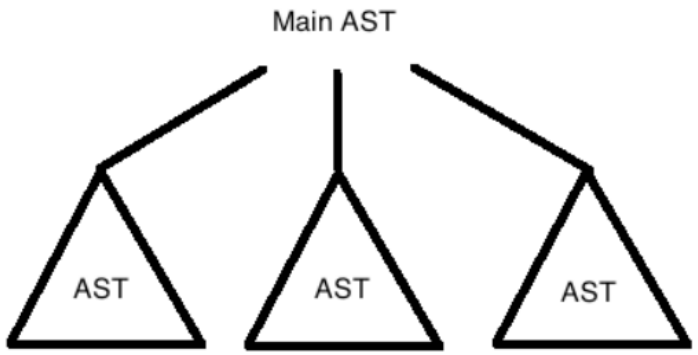
FSCALR files are imported according to TSCALR headers

JAVACC grammar creates token stream, takes semantic actions



<id,1><=><id, 2><+><id, 3>

Back end



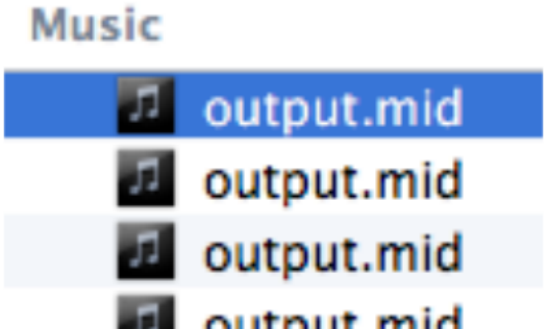

```
func fun(x, y)
    y = []
    while(x++ < 5)
        y += note.pitch(+x)
        if(x % 2 == 0)
            y += note.pitch(+(-x))
        end
    end
    y
end
```

Python Midiutil

Python (2.7+) library that easily makes MIDI from Python, abstracted for our project.

Example input: "[1,2,3|4,5,6][5,4,2]"

Example output: MIDI File!



MIDI

Tools, Processes, Conclusions

Tools

+JavaCC
Github
Python Midiutil
Make

Languages:
Java 1.6
Python 2.7

Process

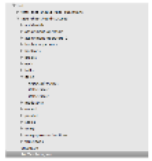
Github repo so that code changes don't conflict.

Built in order of translation strategy, from front-end to back-end.

Contributions on an as-needed basis: who can help who at this moment?

Tests

- Tests written modularly, for each piece of functionality.
- Tests suite run by a suite that color codes the responses (red/green) according to success/failure.



Lessons Learned

- Project Management
- Language Guru
- Architect
- System Integration
- Testing

Tools

Github

Python Midiutil

Make

Languages:

Java 1.6

Python 2.7

Process

Github repo so that code changes don't conflict.

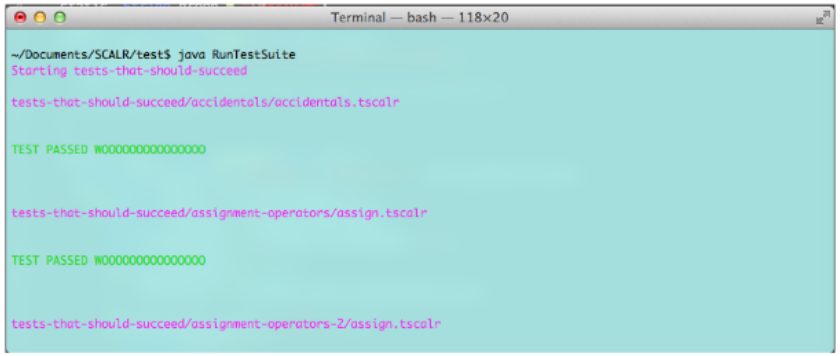
Built in order of translation strategy, from front-end to back-end.

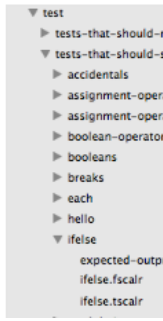
Contributions on an as-needed basis: who can help who at this moment?

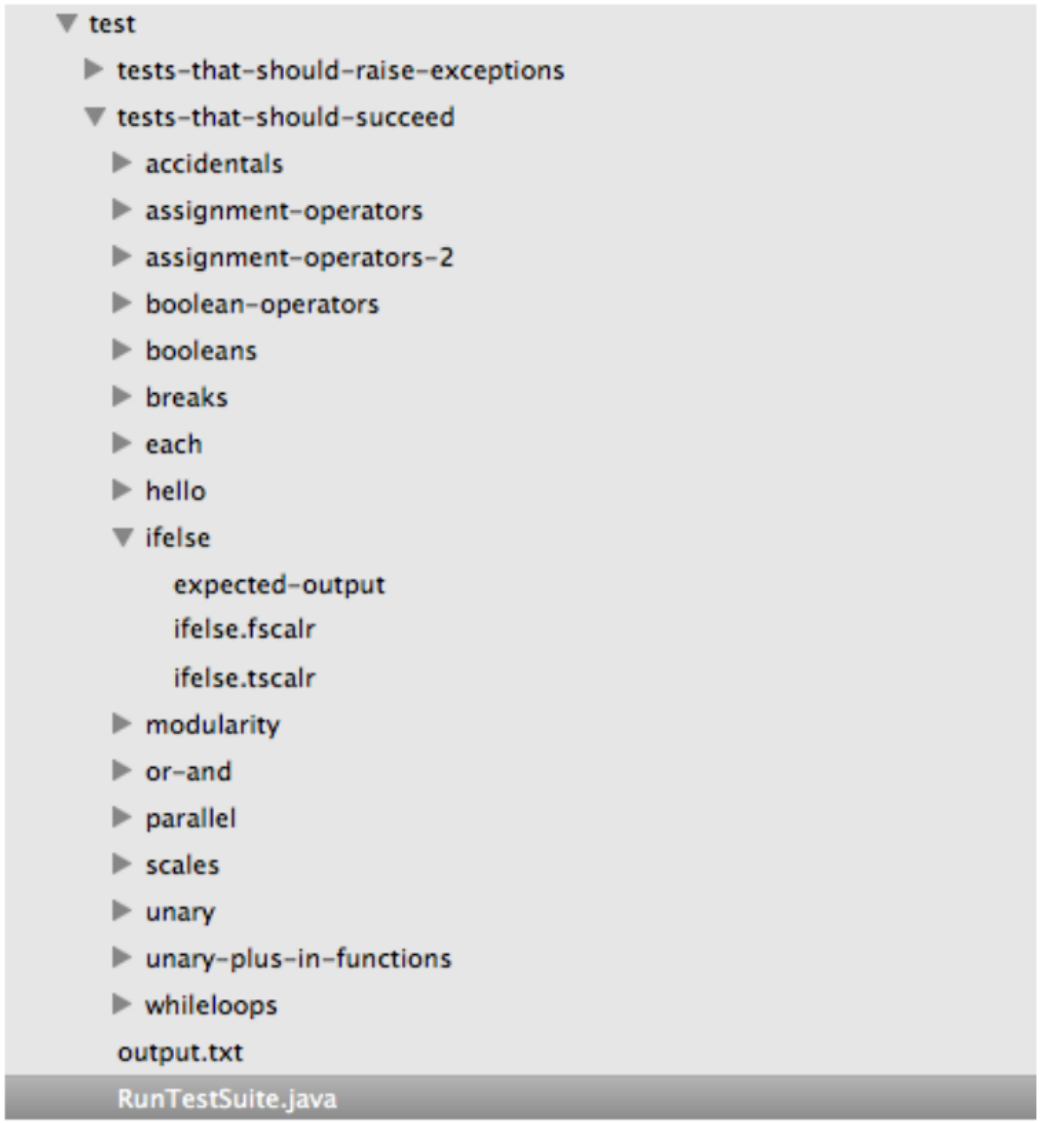
Tests

- Tests written modularly, for each piece of functionality.

- Tests suite run by a suite that color codes the responses (red/green) according to success/failure.






test
tests-that-should-raise-exceptions
tests-that-should-succeed
accidentals
assignment-operators
assignment-operators-2
boolean-operators
booleans
breaks
each
hello
ifelse
expected-output
ifelse.fscalr
ifelse.tscalr
modularity
or-and
parallel
scales
unary
unary-plus-in-functions
whileloops
output.txt
RunTestSuite.java

Lessons Learned

- Project Management
- Language Guru
- Architect
- System Integration
- Testing