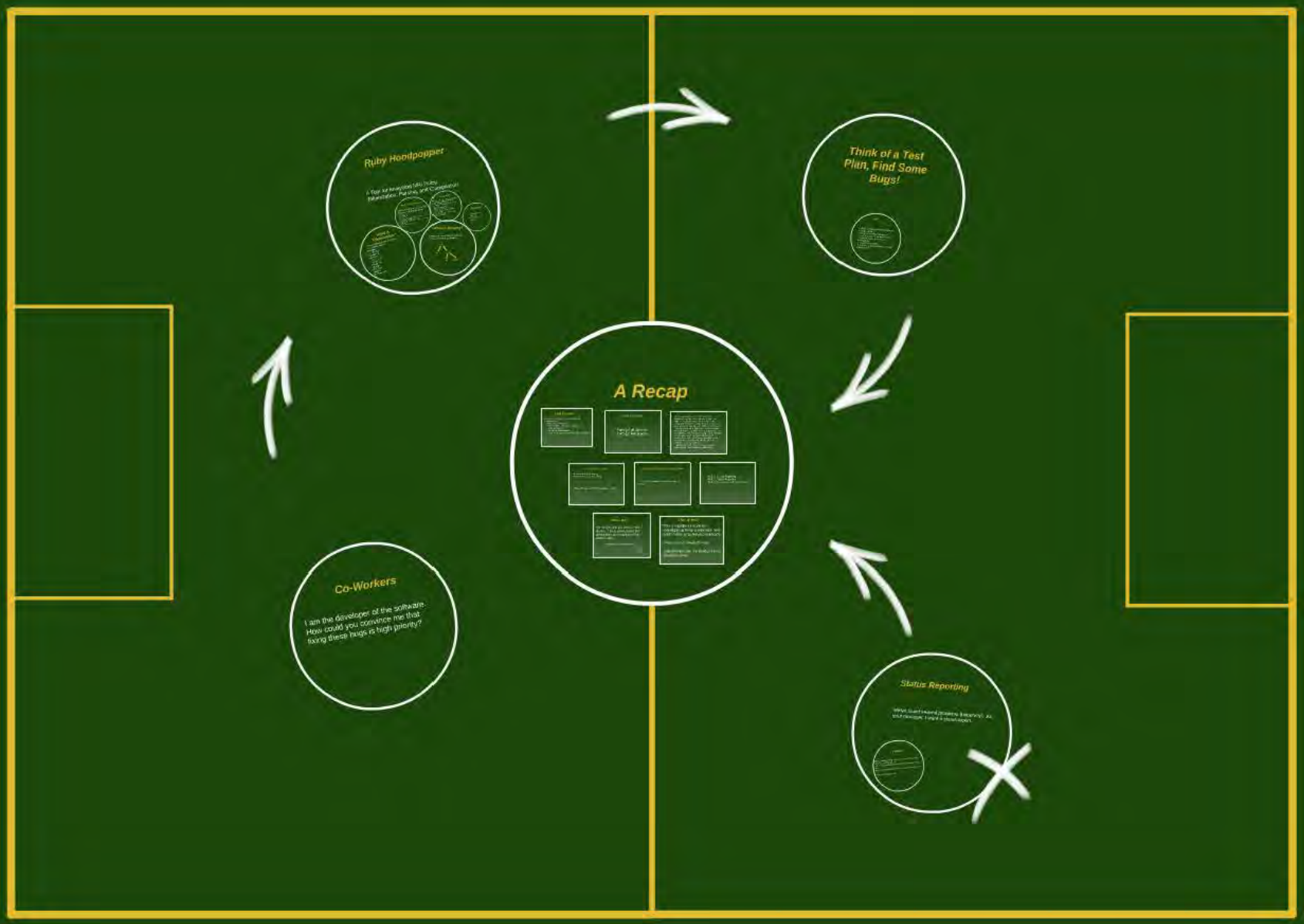
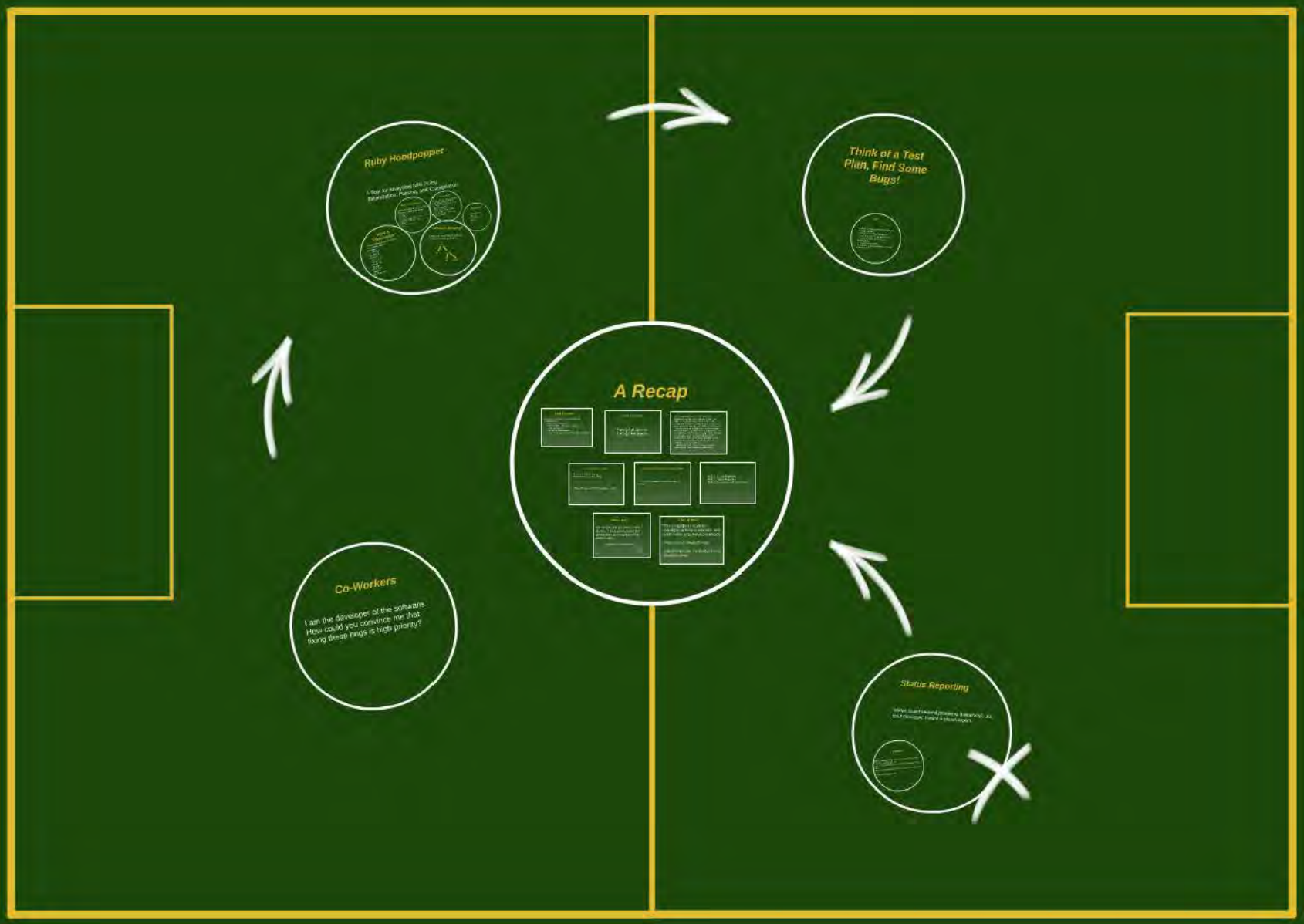
CS1699 - Lecture 11 - Interacting with Stakeholders, Part II



CS1699 - Lecture 11 - Interacting with Stakeholders, Part II



A Recap

Last Lecture...

We discussed general interactions with stakeholders...
- > What is a stakeholder?
- > What should I discuss with them?
- > How to discuss?
- > Clarifying Requirements
- > Understanding and speaking their language(s)

This Lecture

Putting it all together.
Putting it into practice.

"[I]f we take any idea which is abstract or incomplete, we find, on examination, that if we forget its incompleteness, we become involved in contradictions; these contradictions turn the idea in question into its opposite, or antithesis; and in order to escape, we have to find a new, less incomplete idea, which is the synthesis of our original idea and its antithesis. This new idea, though less incomplete than the idea we started with, will be found, nevertheless, to be still not wholly complete, but to pass into its antithesis, with which it must be combined in a new synthesis."
-Bertrand Russell, summarizing Hegelian philosophy in *The Problems of Philosophy*

Practice Makes Perfect

We've talked about testing
We've talked about interacting

Now let's do some interactive testing.

We Are Going To Test My Rails Application

... and you are going to report to me about its quality.

Part 1 - Defect Reporting
Part 2 - Status Reporting
Part 3 - Discussing with Co-Workers

Black Box

For tonight, you are all black box testers. This is open-source but please don't go Googling for the source code.*

* I assume you're all honorable.

This is real!

This is a project I made for investigating Ruby compilation and optimization of generated bytecode.

There are no "seeded" bugs.

Any mistakes are the fault of a poor developer (me).

Last Lecture...

We discussed general interactions with stakeholders..

> What is a stakeholder?
> What should I discuss with them?
> How to discuss?
> Clarifying Requirements
> Understanding and speaking their language(s)

This Lecture

Putting it all together.
Putting it into practice.

"[I]f we take any idea which is abstract or incomplete, we find, on examination, that if we forget its incompleteness, we become involved in contradictions; these contradictions turn the idea in question into its opposite, or antithesis; and in order to escape, we have to find a new, less incomplete idea, which is the synthesis of our original idea and its antithesis. This new idea, though less incomplete than the idea we started with, will be found, nevertheless, to be still not wholly complete, but to pass into its antithesis, with which it must be combined in a new synthesis."

-Bertrand Russell, summarizing Hegelian philosophy in *The Problems of Philosophy*

Practice Makes Perfect

We've talked about testing
We've talked about interacting

Now let's do some interactive testing.

We Are Going To Test My Rails Application

... and you are going to report to me about its quality.

Part 1 - Defect Reporting
Part 2 - Status Reporting
Part 3 - Discussing with Co-Workers

Black Box

For tonight, you are all black box testers. This is open-source but please don't go Googling for the source code.*

* I assume you're all honorable.

This is real!

This is a project I made for investigating Ruby compilation and optimization of generated bytecode.

There are no "seeded" bugs.

Any mistakes are the fault of a poor developer (me).

Ruby Hoodpopper

A Tool for Analyzing MRI Ruby Tokenization, Parsing, and Compilation

What is Compilation?

Takes an AST and converts it to bytecode/machine language.

```
0000 trace 1 ( 1)
0002 putobject 20
0004 putself
0005 opt_send_simple <callinfo!mid:b15, argc:0, FCALL|VCALL|ARGS_SKIP>
0007 opt_plus <callinfo!mid:+, argc:1, ARGS_SKIP>
0009 dup
0010 setlocal_OP__WC__0 2
0012 leave
```

Ruby Math

You don't need to be a Rubyist to test this. You can find some bugs doing some simple math and these rules:
Simple numeric variables don't need to be declared.
Use = for assignment.
Use (), +, -, /, * just like in Java.
Make an arbitrary length array with:
a = Array.new.
Arrays are accessed with [], e.g., a[5].
Use print to display a value.

Example

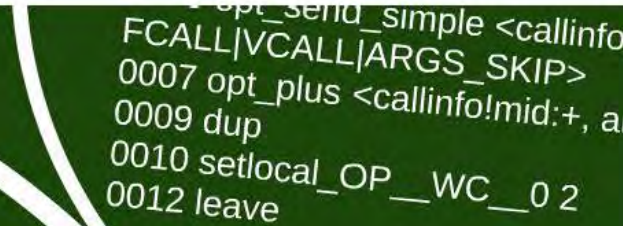
```
a = Array.new
a[3] = 7
b = a[3] + (2 + 1)
print b

10
```

What is Tokenization?

Takes a stream of text and turns it into discrete tokens.
Example:
a = 20 + b15
Identifier: a
<space>
Equals_Sign
<space>
Integer: 20
<space>
Plus_Sign
<space>
Identifier: b15

What is Parsing?

Takes a stream of tokens and turns it into an abstract syntax tree



What is Tokenization?

Takes a stream of text and turns it into discrete tokens.

Example:

a = 20 + b15

Identifier: a

<space>

Equals_Sign

<space>

Integer: 20

<space>

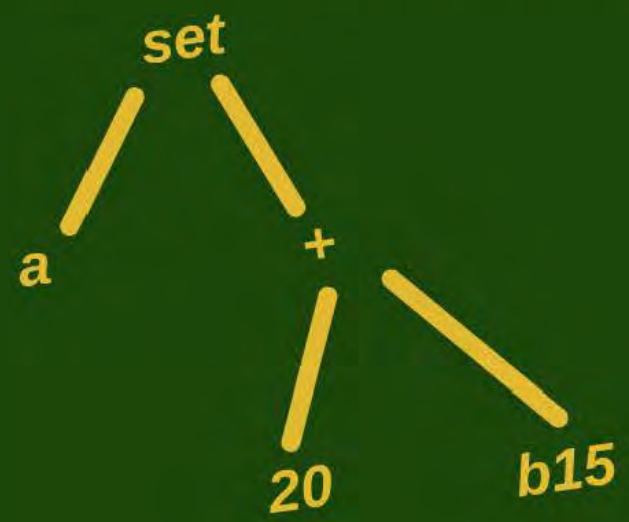
Plus_Sign

<space>

Identifier: b15

What is Parsing?

Takes a stream of tokens and turns it into an abstract syntax tree



What is Compilation?

Takes an AST and converts it to bytecode/machine language.

```
0000 trace 1 ( 1)
0002 putobject 20
0004 putself
0005 opt_send_simple <callinfo!mid:b15, argc:0,
FCALL|VCALL|ARGS_SKIP>
0007 opt_plus <callinfo!mid:+, argc:1, ARGS_SKIP>
0009 dup
0010 setlocal_OP__WC__0 2
0012 leave
```

You do
You ca
simple
Simple
declare
Use =
Use (),
Make a
a = Arr
Arrays
Use pr

n?

Wha

t to

c:0,

_SKIP>

Ruby Math

You don't need to be a Rubyist to test this.
You can find some bugs doing some simple math and these rules.
Simple numeric variables don't need to be declared.
Use = for assignment.
Use (), +, -, /, * just like in Java.
Make an arbitrary length array with:
a = Array.new.
Arrays are accessed with [], e.g., a[5].
Use print to display a value.

Example

```
a = Array.new
a[3] = 7
b = a[3] + (2 + 1)
print b
```

10

g?

Think of a Test Plan, Find Some Bugs!

Hints:

1. What are the steps where something could go wrong?
2. What are the important steps?
3. Can you think of enhancements?
4. Is there anything difficult to understand?
5. What is the severity?
6. Can you cause problems externally (client-side)?

Hints:

1. What are the steps where something could go wrong?
2. What are the important steps?
3. Can you think of enhancements?
4. Is there anything difficult to understand?
5. What is the severity?
6. Can you cause problems externally (client-side)?

A Recap

Last Lecture...

We discussed general interactions with stakeholders...
- > What is a stakeholder?
- > What should I discuss with them?
- > How to discuss?
- > Clarifying Requirements
- > Understanding and speaking their language(s)

This Lecture

Putting it all together.
Putting it into practice.

"[I]f we take any idea which is abstract or incomplete, we find, on examination, that if we forget its incompleteness, we become involved in contradictions; these contradictions turn the idea in question into its opposite, or antithesis; and in order to escape, we have to find a new, less incomplete idea, which is the synthesis of our original idea and its antithesis. This new idea, though less incomplete than the idea we started with, will be found, nevertheless, to be still not wholly complete, but to pass into its antithesis, with which it must be combined in a new synthesis."
-Bertrand Russell, summarizing Hegelian philosophy in *The Problems of Philosophy*

Practice Makes Perfect

We've talked about testing
We've talked about interacting

Now let's do some interactive testing.

We Are Going To Test My Rails Application

... and you are going to report to me about its quality.

Part 1 - Defect Reporting
Part 2 - Status Reporting
Part 3 - Discussing with Co-Workers

Black Box

For tonight, you are all black box testers. This is open-source but please don't go Googling for the source code.*

* I assume you're all honorable.

This is real!

This is a project I made for investigating Ruby compilation and optimization of generated bytecode.

There are no "seeded" bugs.

Any mistakes are the fault of a poor developer (me).

Status Reporting

We've found several problems (hopefully). As your manager, I want a status report.

Think:

What is the best way to subdivide the functionality?
How bad are the defects?
What would be the best way to communicate this to me?
How important would it be to add the enhancements?

Use your best judgment.

Think:

What is the best way to subdivide the functionality?
How bad are the defects?
What would be the best way to communicate this to me?
How important would it be to add the enhancements?

Use your best judgment.

Co-Workers

I am the developer of the software.
How could you convince me that
fixing these bugs is high priority?

CS1699 - Lecture 11 - Interacting with Stakeholders, Part II