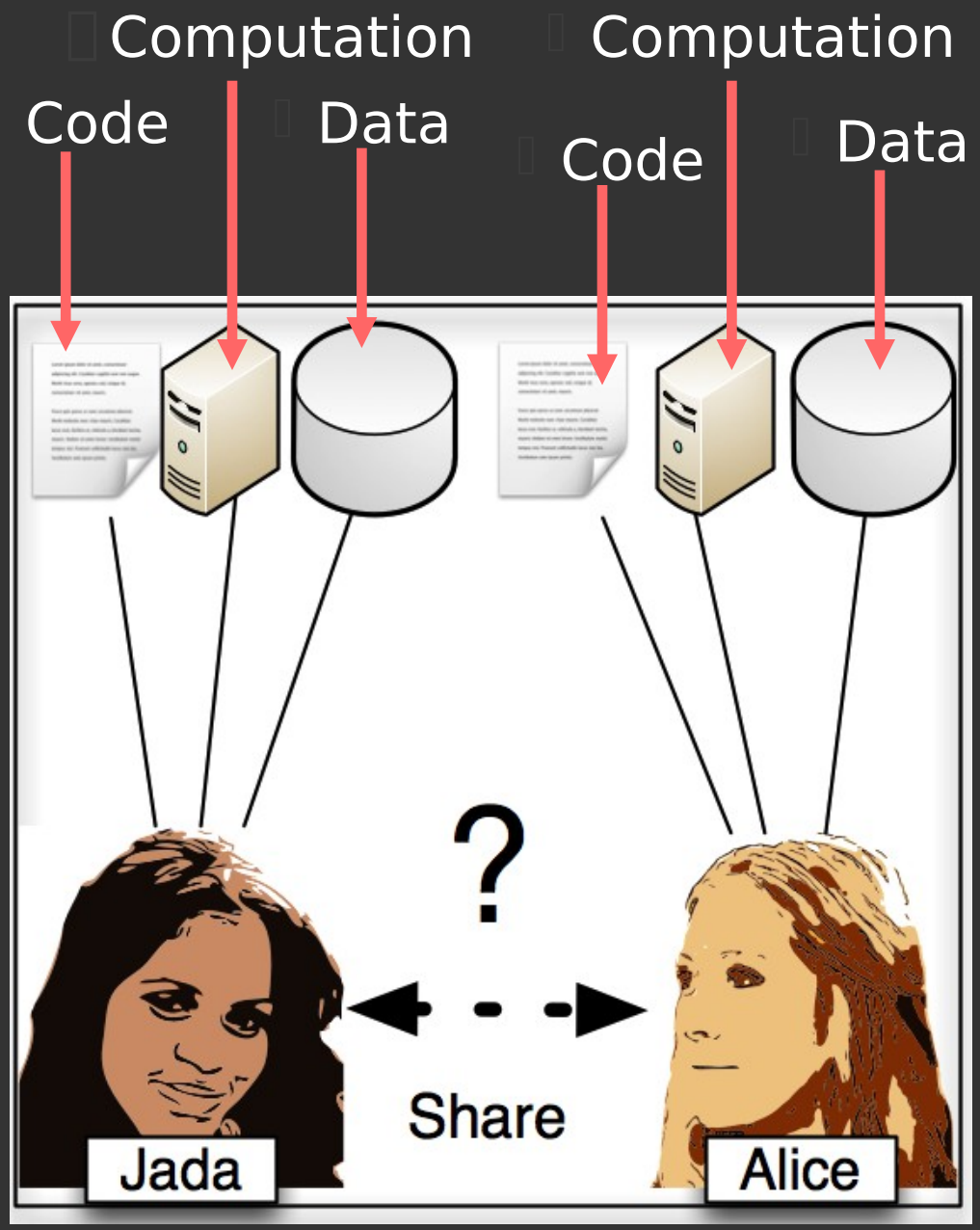
The Graphyte Project
design, tools and practices

Speaker: Yeukhon Wong
Michael Grossberg, Irina Gladkova,
Jeremy Neiman, Hannah Aizenman
GLASS Lab, CUNY City

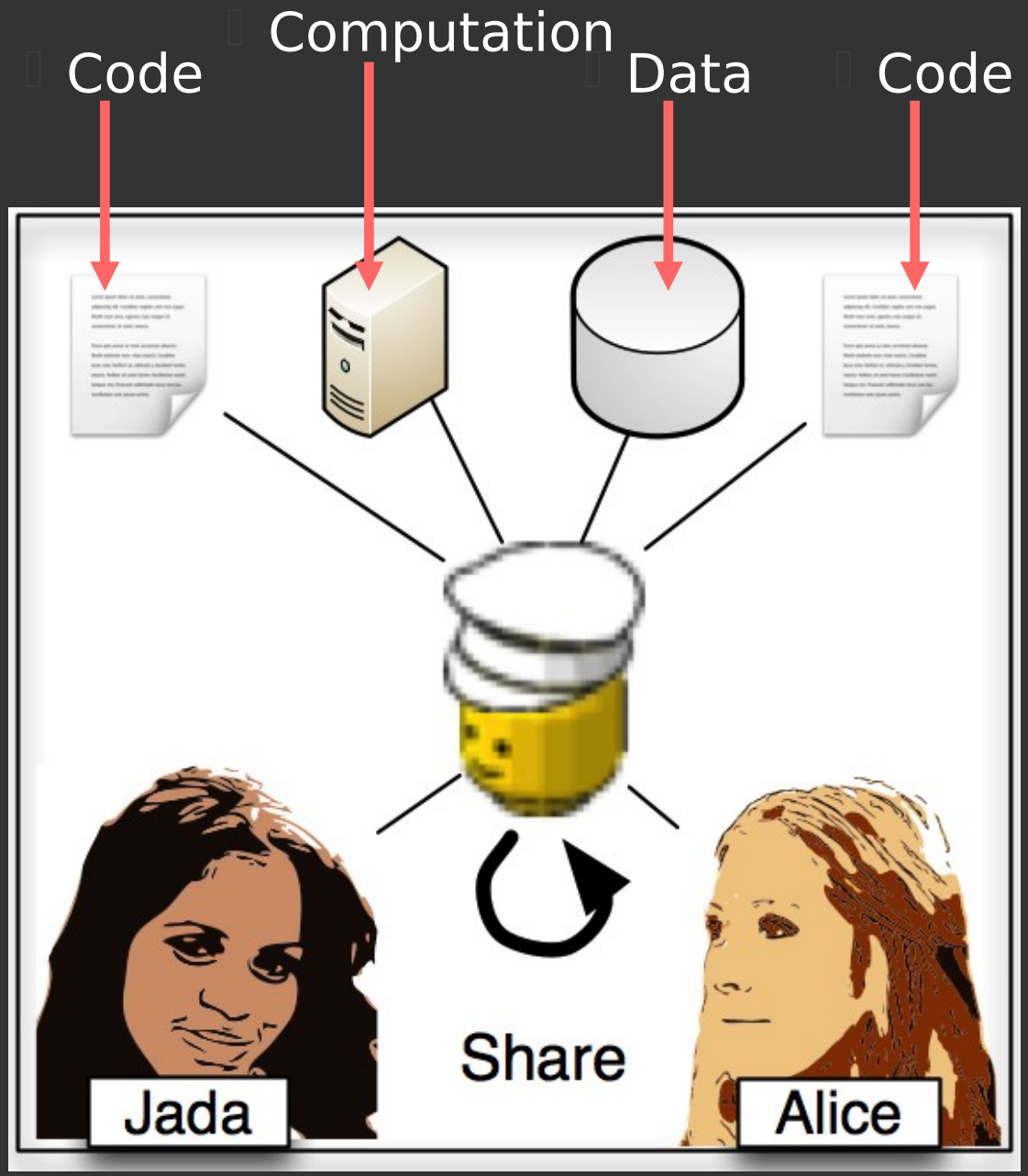
April 1st, 2013
SEA Software Engineering Conference 2013


C
6
CARBON
12.01
2.26

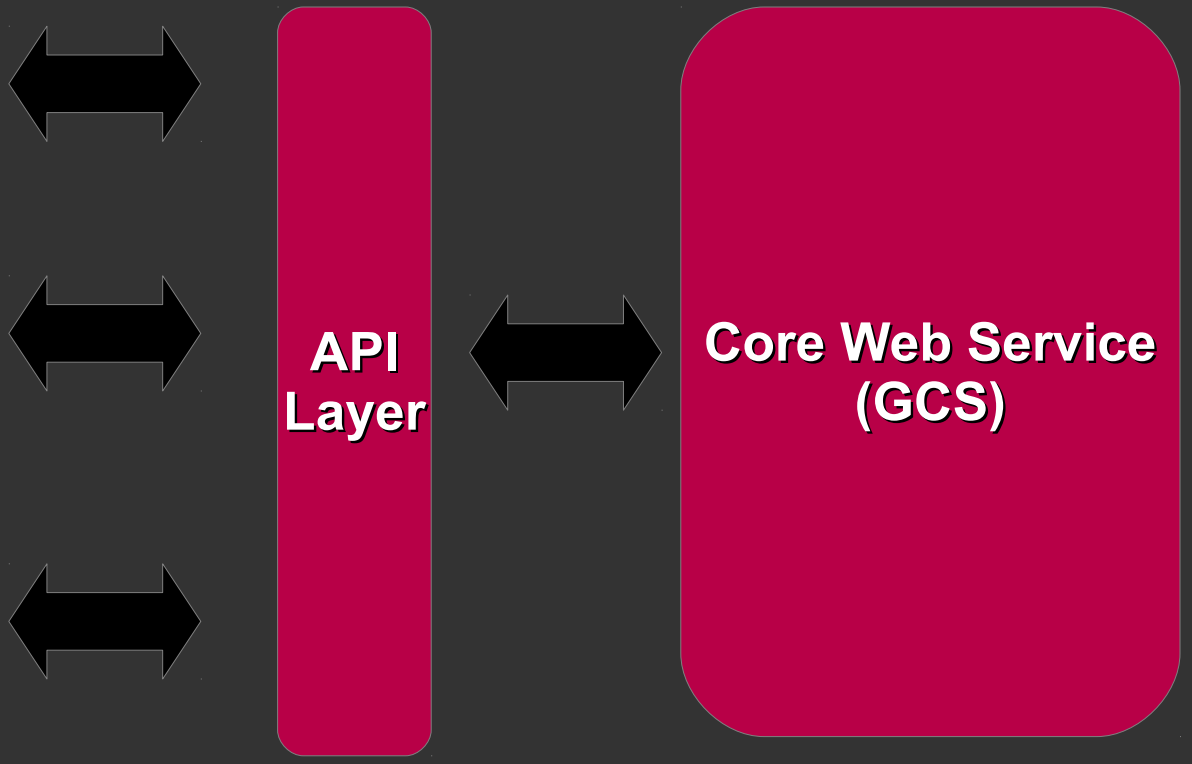
Traditional Workflow



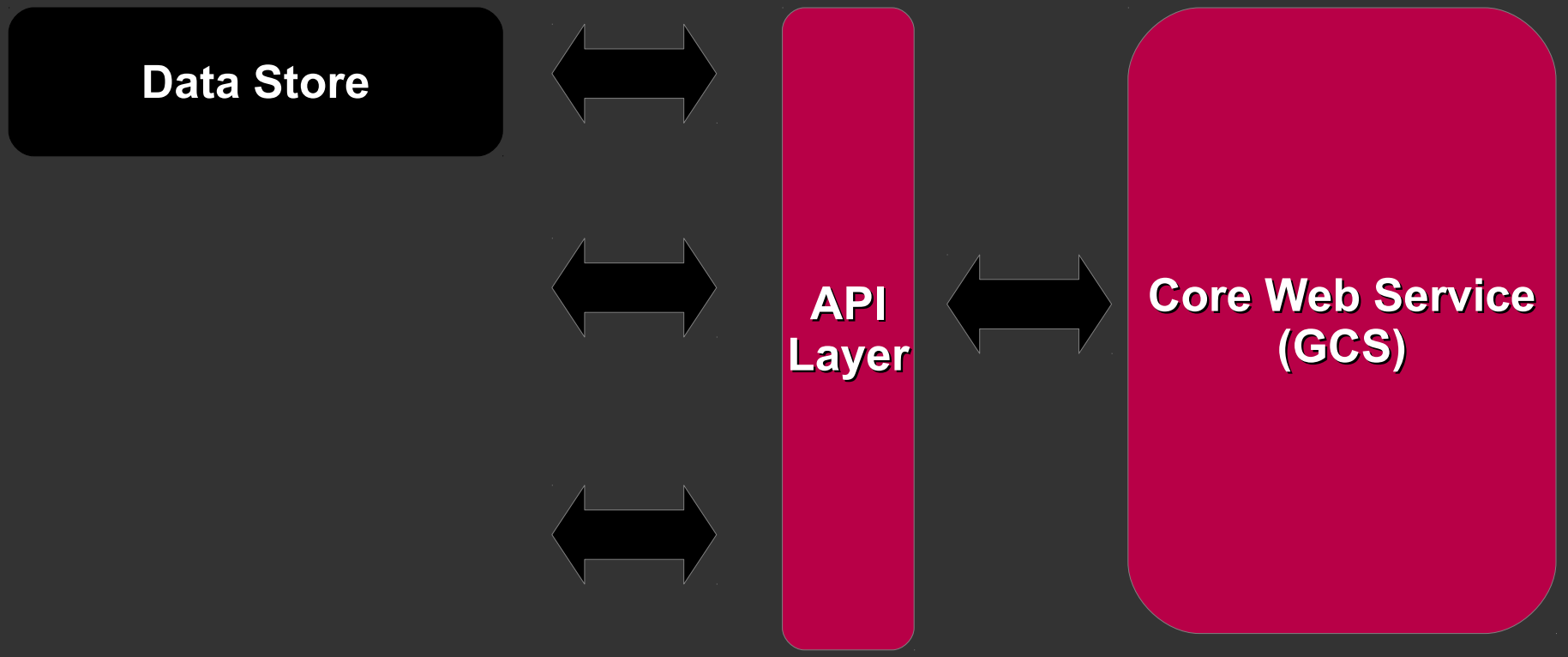
Ideal Workflow



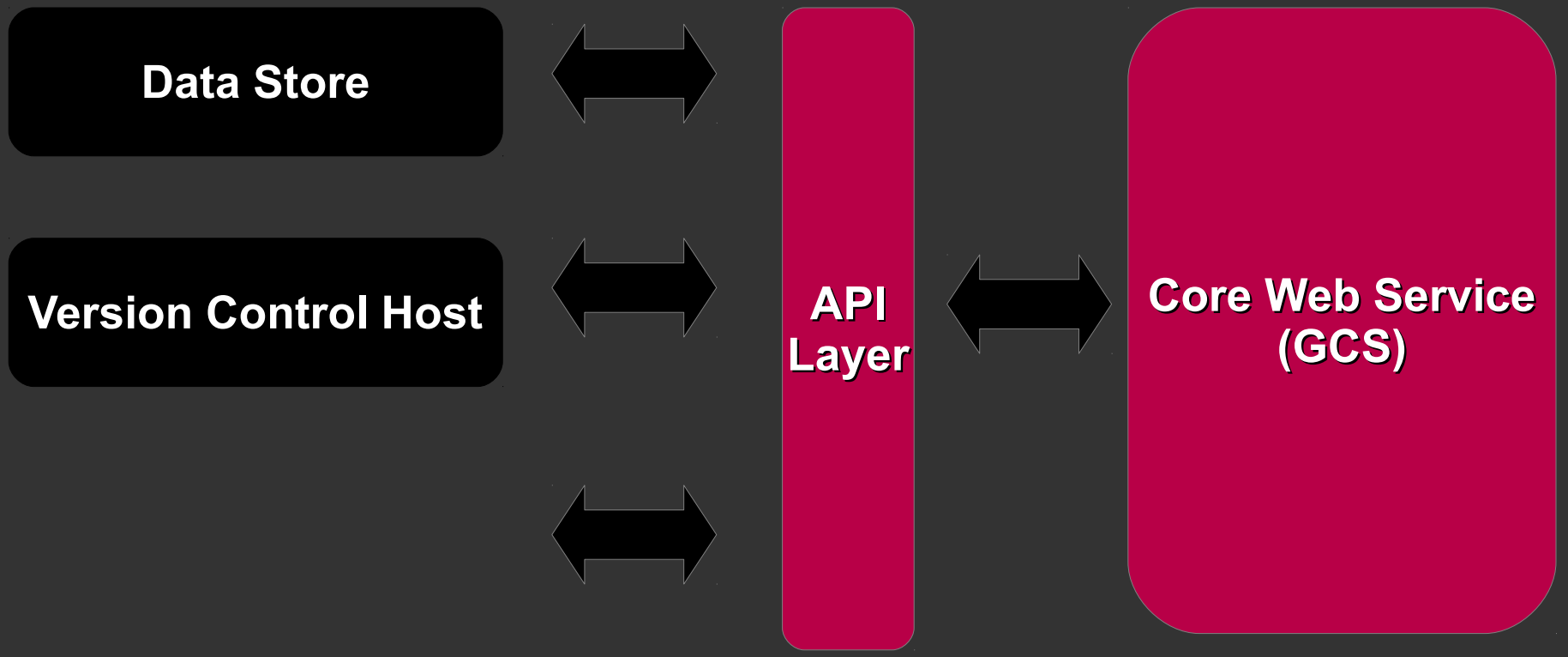
Graphyte Architecture



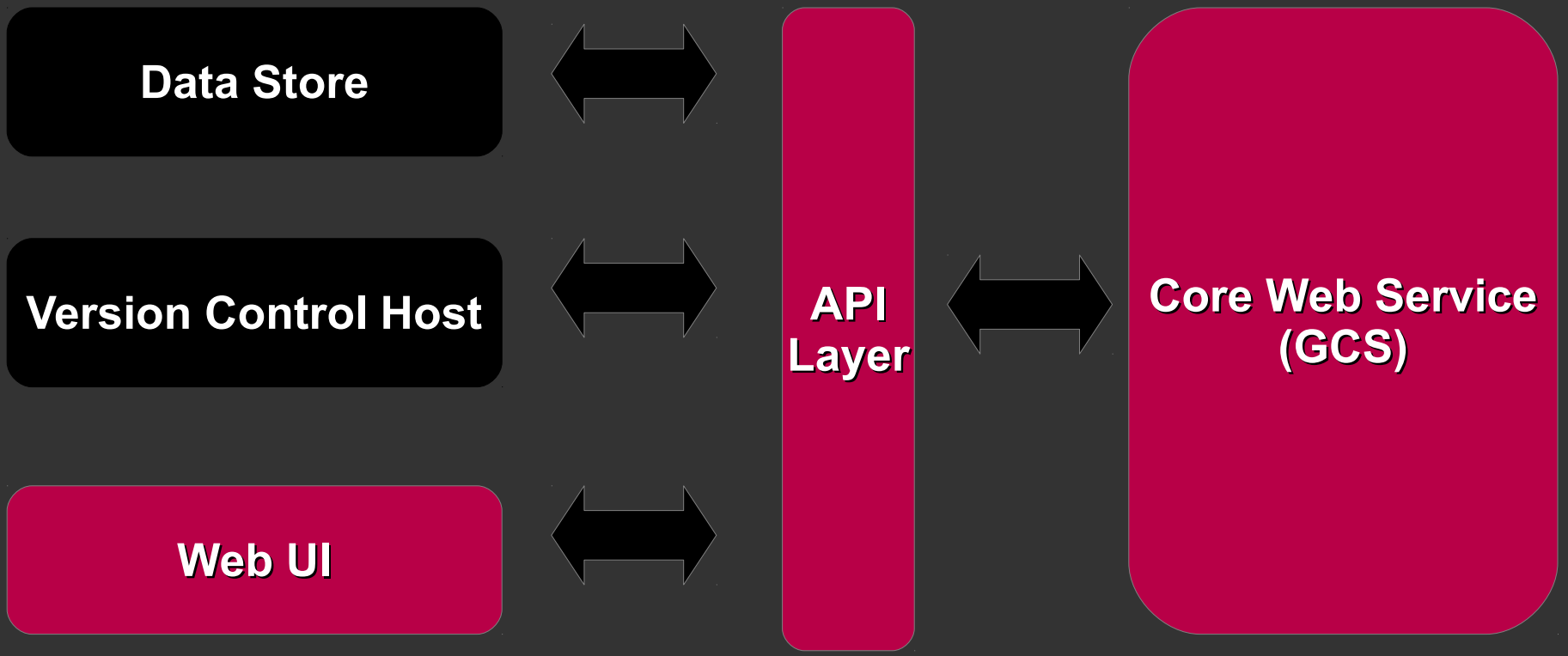
Graphyte Architecture



Graphyte Architecture



Graphyte Architecture

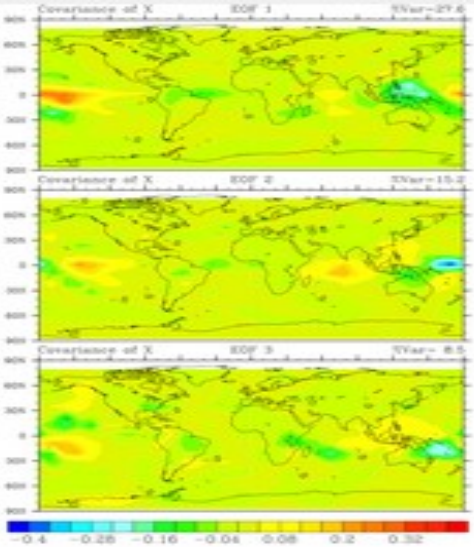


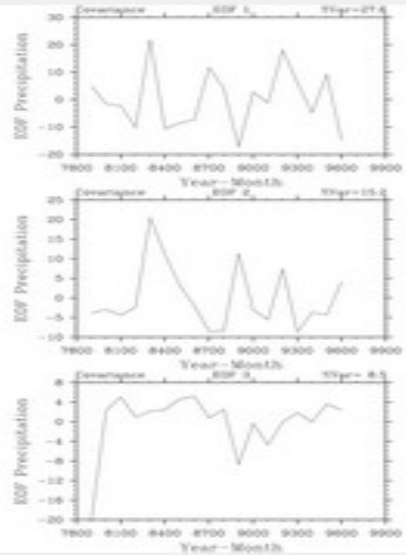
Web UI: Result HIstory

Run history for climate_eof_analysis

Start Time	Status	Run Type	Actions
2012-11-13 13:06:31.887932	Error	Shell Script	View Details
2012-11-13 14:37:49.538312	Error	Shell Script	View Details
2012-11-06 18:24:19.175156	Error	Shell Script	View Details
2012-11-06 15:07:16.260062	Error	Shell Script	View Details
2012-11-06 15:05:36.288337	Complete	Shell Script	View Details
2012-11-06 15:03:06.912129	Error	Shell Script	View Details
2012-10-24 12:26:45.480515	Error	Shell Script	View Details

artifacts





- climo.000001.png
- climo.000002.png

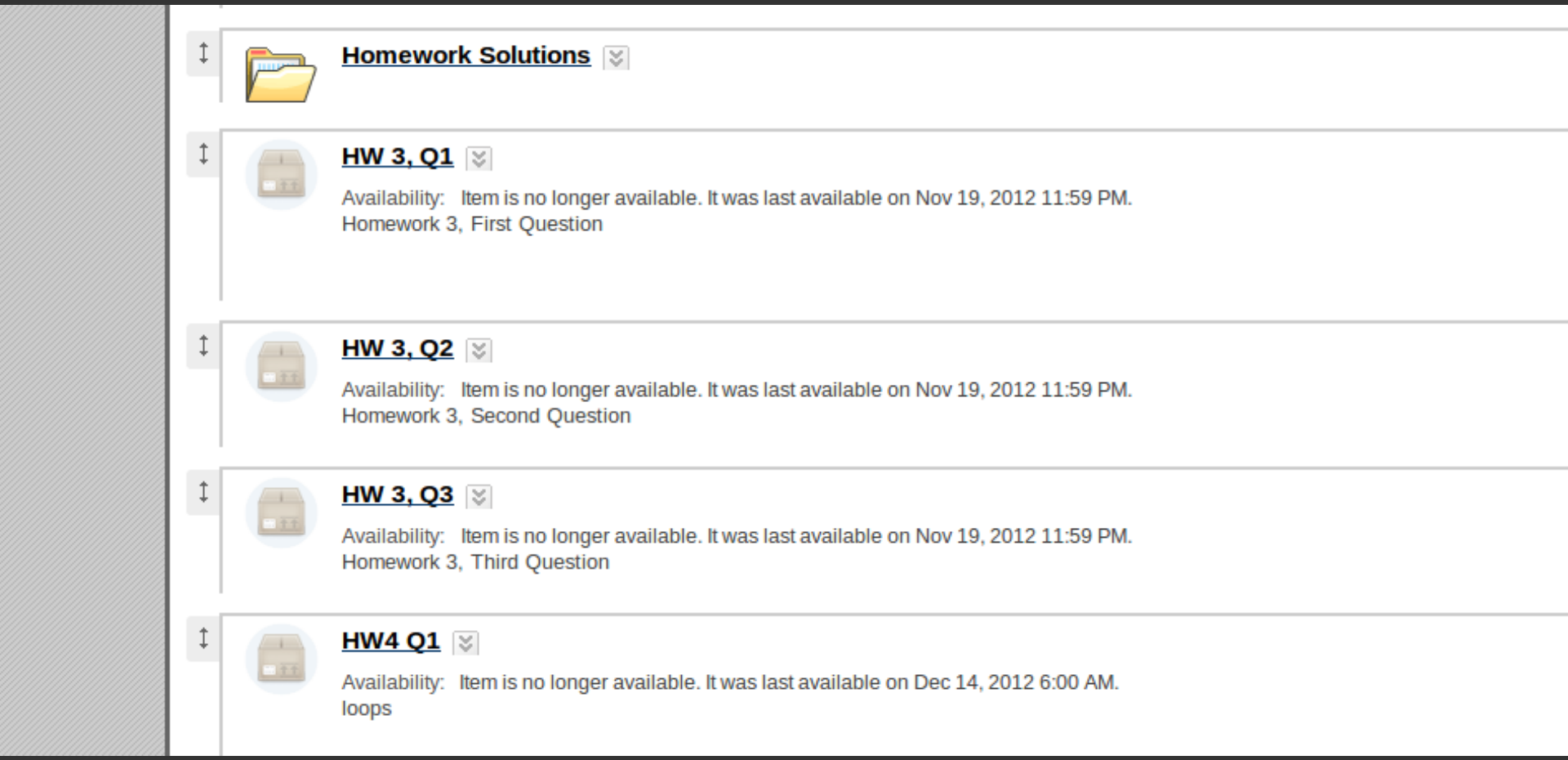
Web UI: Result Details


Au 79
GOLD
197.0 19.3
1064 2808

Aurum

- A RESTful web service built using Graphyte API and infrastructure.
- Use automated tests to evaluate learning progress.

Aurum on Blackboard



List of prime numbers.

Problem: Generating Primes

In this problem you will write a Python program to generate a list of all the the prime numbers between a starting number and an ending number (inclusive).

Background

A prime number is any whole number greater than 1 that is divisible only by itself and 1. Prime numbers are important in mathematics and computer science because they are used in many theorems and algorithms. Prime numbers are very commonly used in cryptography (the field of computer science devoted to keeping information secret), so there are a number of methods for generating prime numbers on the fly. Many of these methods are called sieves, because they generate a list of possible primes and then filter out the numbers that are not primes. Generating Primes

Problem

Problem description

1. Repeatedly ask for inputs until the start and stop numbers are greater than 0.

Problem instruction

```
Enter 2 positive numbers:
-1 1
Enter 2 positive numbers:
0 0
Primes between 0 and 0:
```

2. Print out a list of all prime numbers between the starting and stop numbers.

```
Enter 2 positive numbers: 3 7
Primes between 3 and 7: 3 5 7
```

3. Make sure it works regardless of input order.

```
Enter 2 positive numbers: 7 3
Primes between 3 and 7: 3 5 7
```

Make sure it works for arbitrary start and stop inputs:

```
Enter 2 positive numbers: 10 30
Primes between 10 and 30: 11 13 17 19 23 29
```

Tips on using this learning module:

- Do not close this windows until the submit grade button at the end of the page is re-enabled.
- If you close this windows during grading, you may receive a zero. Do not panic! You can always come back and submit the grade later.
- If you haven't accepted the SSL certificate yet, please do so by right click and open this link as new tab, accept the certificate and reload the exercise. (help video: chrome / firefox)

Enter your code:

User input (code, multiple-choice, etc)

```
#include <iostream>
#include <cmath>

using namespace std;

int main(){
   int a,b, tmp;

   do{
      cout<<"Enter 2 positive numbers: ";
      cin>>a>>b;
   }while(a<0 || b<0);

   if (b<a){
      tmp = b;
      b = a;
      a = tmp;
   }

   cout<<"Primes between "<<a<<" and "<<b<<": ";
   int i;
   a = a-1;
   while(a++<b){
      for (i=2;i<a;i++){
```

Grade it

Score Table

Test 0	Success	test_binary_exists	Does the program compile?		0 out of 0
Test 1	Success	test_primes_false	Correctly identifies non-prime numbers.		15 out of 15
Test 2	Success	test_primes_less_n	Prints all primes between a and b.		10 out of 10
Test 3	Success	test_primes_true	Correctly identifies prime numbers.		15 out of 15
Test 4	Success	test_swap_input_order	makes sure the smaller number comes first.		5 out of 5
Test 5	Success	test_valid_pos_input	Checks that the input is positive.		5 out of 5
		Total Score			50 out of 50

Submit Grade

Student Views: automated assessment

Fall 2012 Intro Computing: CSC 10200 G [0524] (CCNY)
Home Page
Information
Content
Discussions
Groups
Tools
Help

COURSE MANAGEMENT

Control Panel
Files
Course Tools
Evaluation
Grade Center
Needs Grading
Full Grade Center
Assignments
Tests
Users and Groups
Groups
Users
Customization
Packages and Utilities
Help

Grade Center : Full Grade Center

In the Screen Reader mode, the table is static and grades may be entered on the Grade Details page accessed by selecting the table cell for the grade. In the interactive mode of the Grade Center, grades can be typed directly in the cells. Use the arrow keys or the tab key to navigate through the Grade Center and the Enter key to submit a grade. More Help

Create Column | Create Calculated Column | Manage | Reports | Filter | Work Offline

Move To Top | Email | Sort Columns By: Layout Position | Order: Ascending

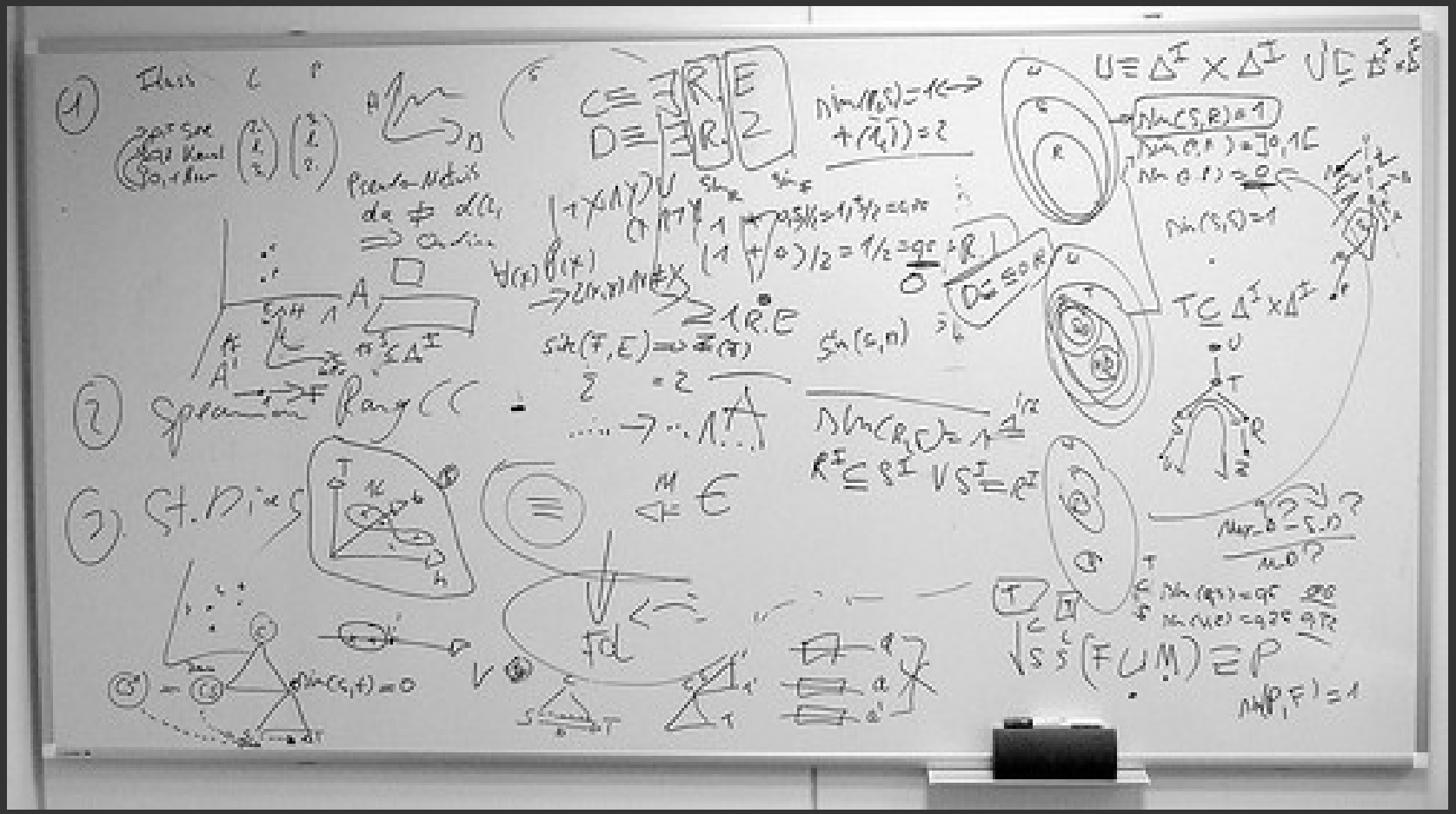
Grade Information Bar | Last Saved:November 22, 2012 12:13 AM

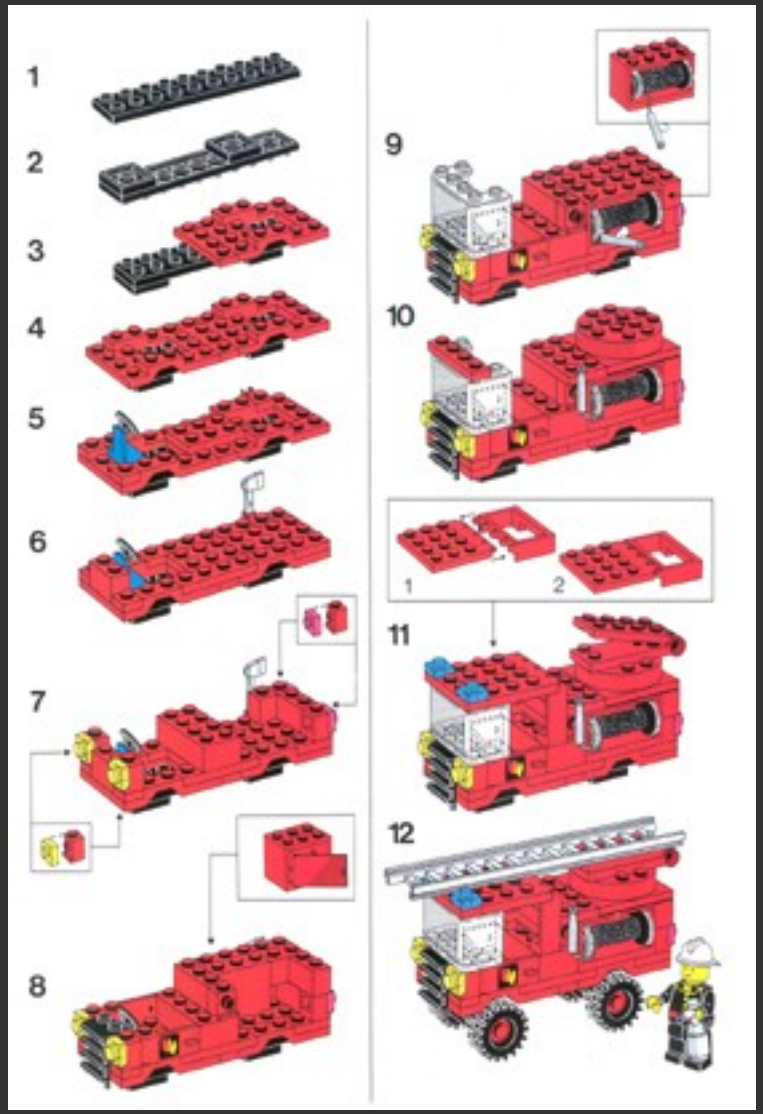
Last Name	First Name	HW 1	HW2	Home	HW3 Q1	HW3 Q2	HW3 Q3
[redacted]	[redacted]	50.00	!	!	15.00	10.00	25.00
[redacted]	[redacted]	36.00	!	!	15.00	10.00	25.00
[redacted]	[redacted]	48.00	!	!	15.00	10.00	25.00
[redacted]	[redacted]	0.00	!	!	15.00	10.00	25.00
[redacted]	[redacted]	0.00	0.00	--	15.00	6.00	0.00
[redacted]	[redacted]	41.00	!	!	15.00	10.00	25.00
[redacted]	[redacted]	43.00	!	!	15.00	10.00	20.50
[redacted]	[redacted]	43.00	!	!	15.00	10.00	25.00
[redacted]	[redacted]	33.00	!	!	10.00	10.00	18.18
[redacted]	[redacted]	48.00	!	!	15.00	10.00	25.00
[redacted]	[redacted]	35.00	!	!	15.00	10.00	18.18
[redacted]	[redacted]	7.00	!	!	15.00	10.00	25.00
[redacted]	[redacted]	45.00	!	!	15.00	10.00	25.00
[redacted]	[redacted]	15.00	!	!	15.00	10.00	20.50 !
[redacted]	[redacted]	44.00	!	!	15.00	10.00	25.00
[redacted]	[redacted]	46.00	!	!	15.00	10.00	25.00
[redacted]	[redacted]	42.00	!	!	15.00	6.70	0.00

Selected Rows: 0

Move To Top | Email | Icon Legend

Lessons

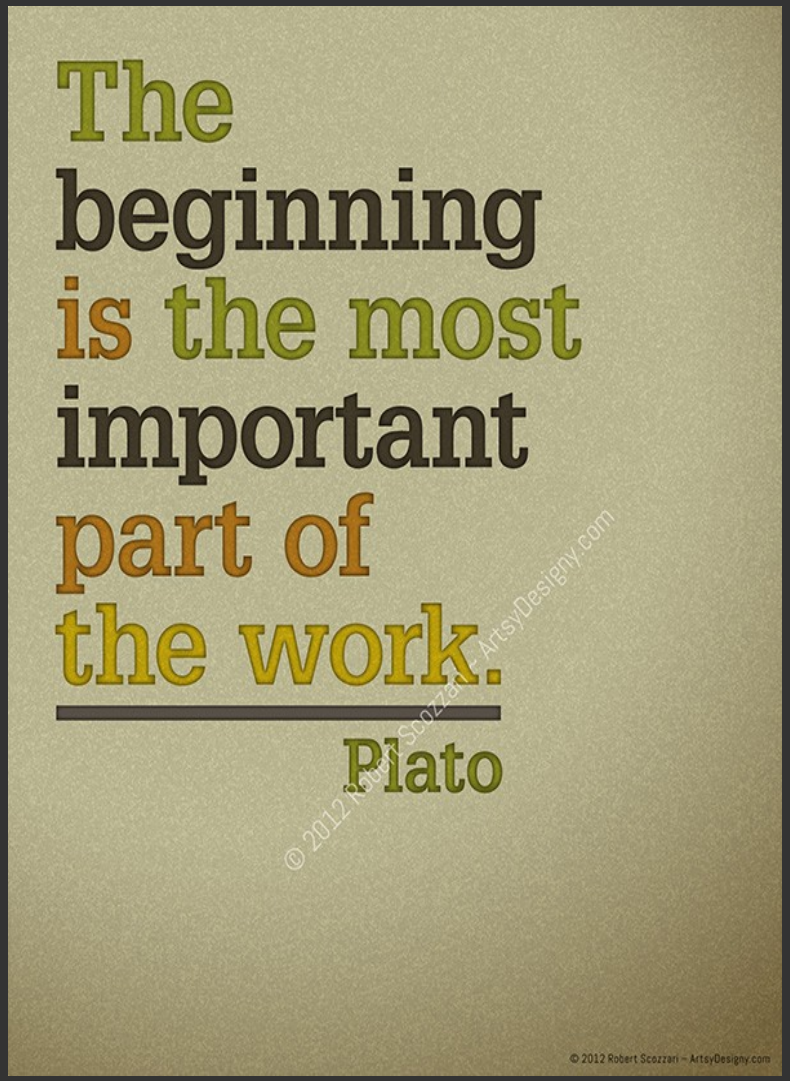




Good:
Modular
Architecture

Bad:
Monolithic
Implementation

Fix: Enforce modularity from beginning



Good:
Web
service-centric

Bad:
Inconsistent API

Fix:

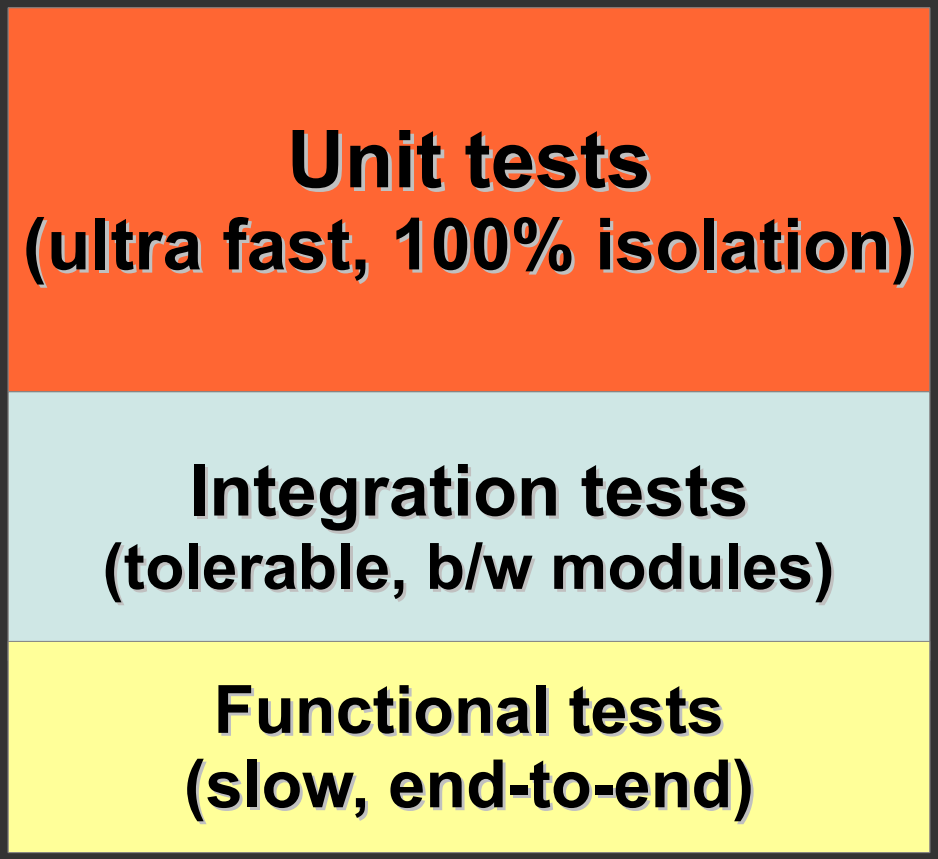
RESTful Patterns

Good:
Automated tests

Bad:

Tests take >1.5hrs

Fix: Write cheap tests



Even for prototyping, use these good practices.

Packaging Servers

Good:

Had instructions to configure machines.

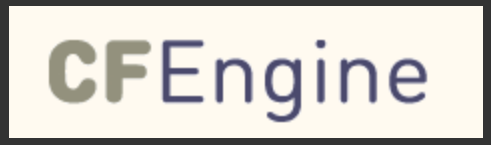
Bad:

No automation,
outdated Wiki

Fix:

Use automated deployment tool

Many tools, which to use?



What works for us

http://ansible.cc/
Ansible
Configure servers with SSH.

1. More tests, more coverage.

2. Write user and developer documentation.

3. Bring back continuous integration build and test.

4. We want VM with roles.

(database servers, web servers, cache server)

5. Open-source Graphyte!

Contributors:

Jeremy Neiman, Hannah Aizenman,

Luis Bello, Yuriy Steijko, Paul K. Alabi,

Joesan Gabaldon, Michael Grossberg,

Irina Gladkova

Questions?

yeukhon@acm.org

glasslab.org