

# Satsy

Input-Output Code Search Engine

*Carl Chapman*  
*Cole Groff*  
*Cody Hoover*  
*Trevor Lund*  
*Kaitlin McAbee*

# Problem Statement

May14  
13



- Lots of open-source code is available for use.
- Instead of rewriting code that already exists, can we search to find the existing code that does what we want?
- Assume we can describe functionality as an Input-Output pair.

# Satsy Example

## *Satsy*

an example-based search engine for source code

Input



Search

Advanced Search

Output

# Google Example



java program determine leap year



**Web**

Videos

Images

Shopping

News

More ▾

Search tools

About 48,200 results (0.48 seconds)

## Java Code for calculating Leap Year - Stack Overflow

[stackoverflow.com/.../java-code-for-calculating-leap-year](#) ▾ Stack Overflow ▾

Jun 20, 2009 - ... "The Art and Science of Java" book and it shows how to **calculate** a **leap year**. ... `public void run() { println("This program calculates leap year.";`

## Java Program to Check whether a given year is a Leap Year...

[www.icsejava.com/programs/leap-year](#) ▾

Mar 13, 2014 - A **leap year** is a year which is divisible by 4, with the exception that if the year is divisible by 100, then it should also be divisible by 400.

# Prototype Market Study

May14  
13

Google

vs.

Satsy

48.4

Results  
Returned

20.5

---

1.5

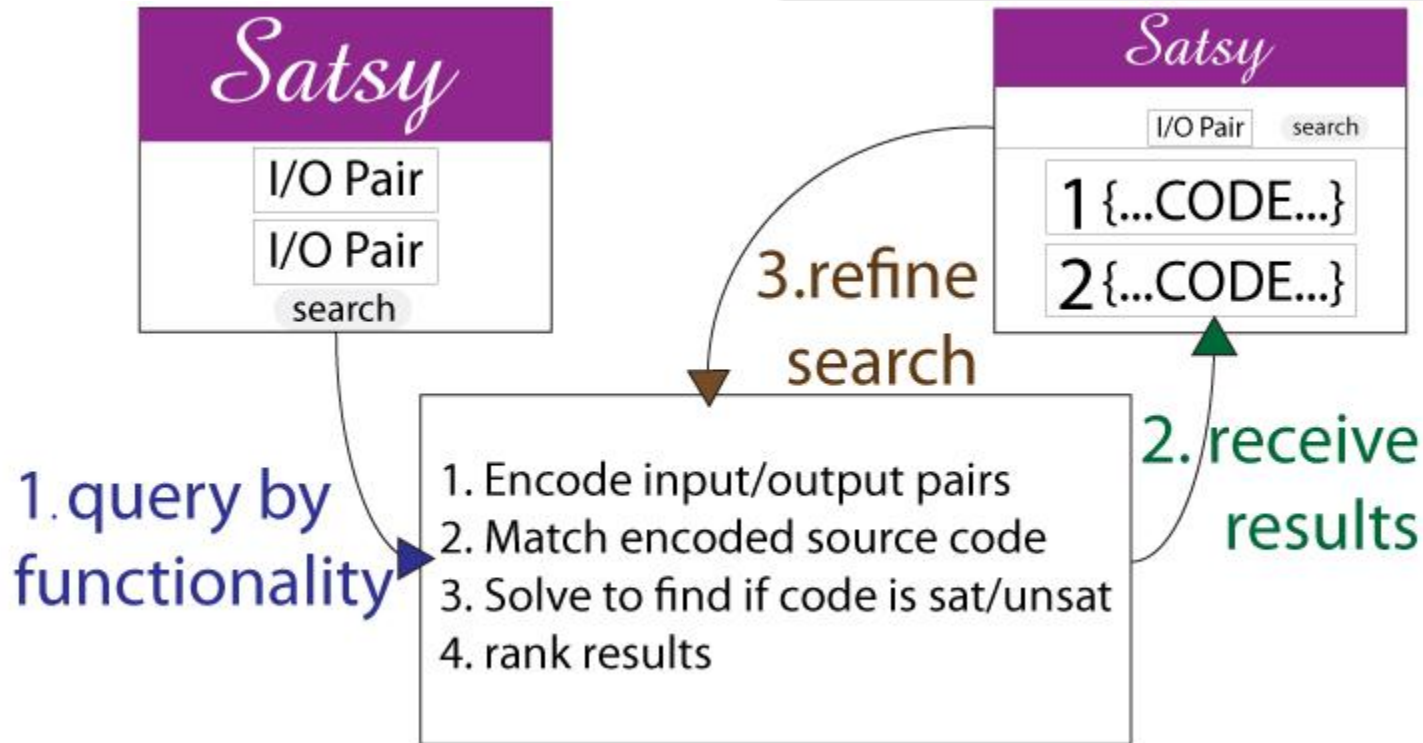
Satisfactory  
Results

8.5

[1]

# Concept Sketch

May14  
13

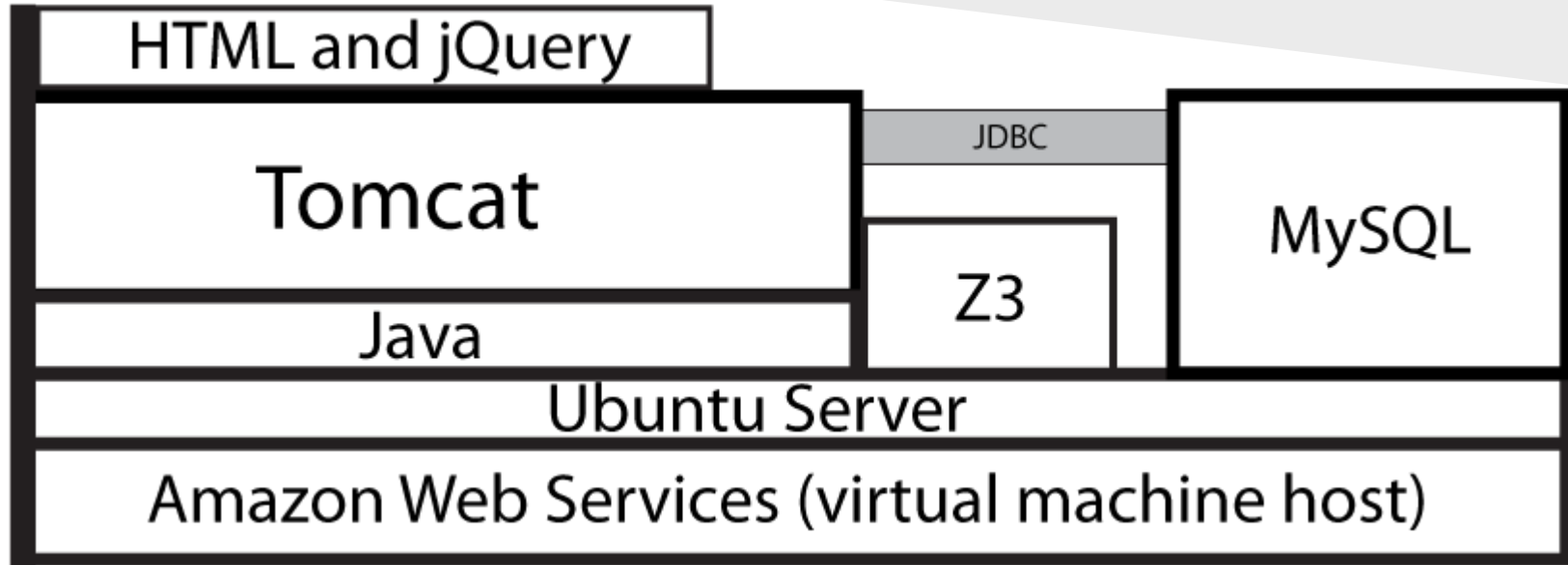


# Live Demonstration

May 14  
**13**

# Challenge 1: Technologies

May14  
13



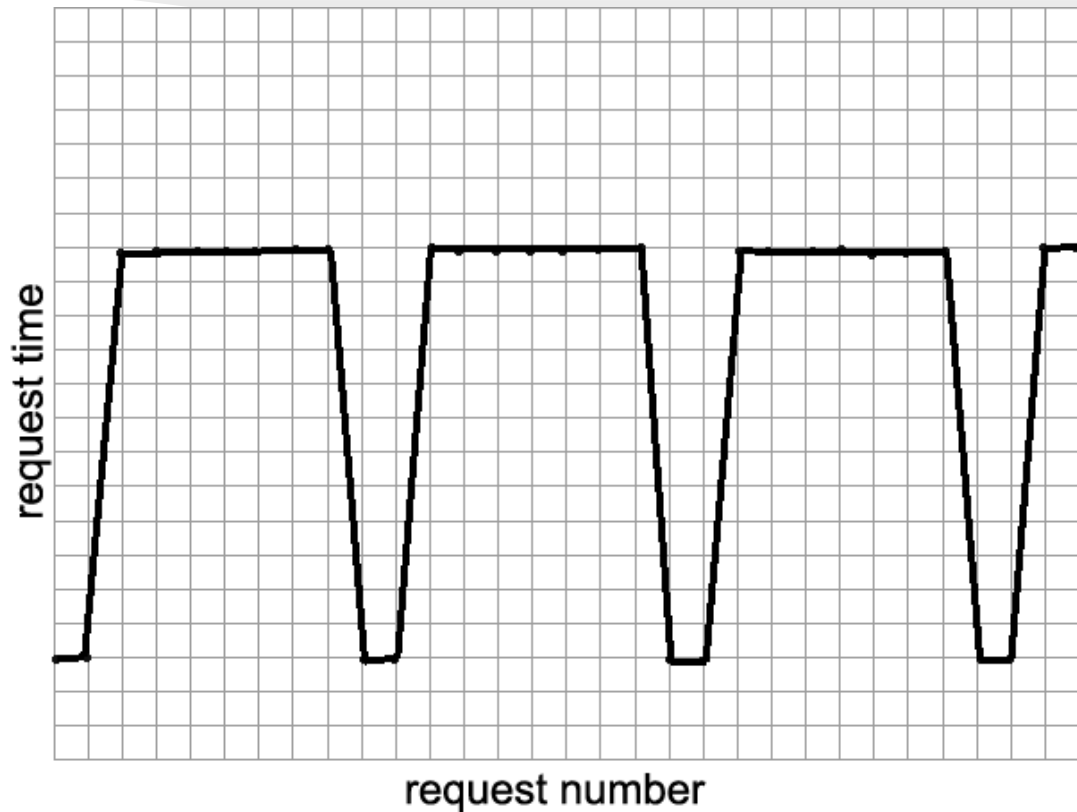
- Amazon Web Services setup
- Running Z3 from Java application
- JDBC connection pool
- Running Java code from HTTP requests
- Asynchronous requests and responses
- Deploying Java web application



# Challenge 2: AWS

May 14  
13

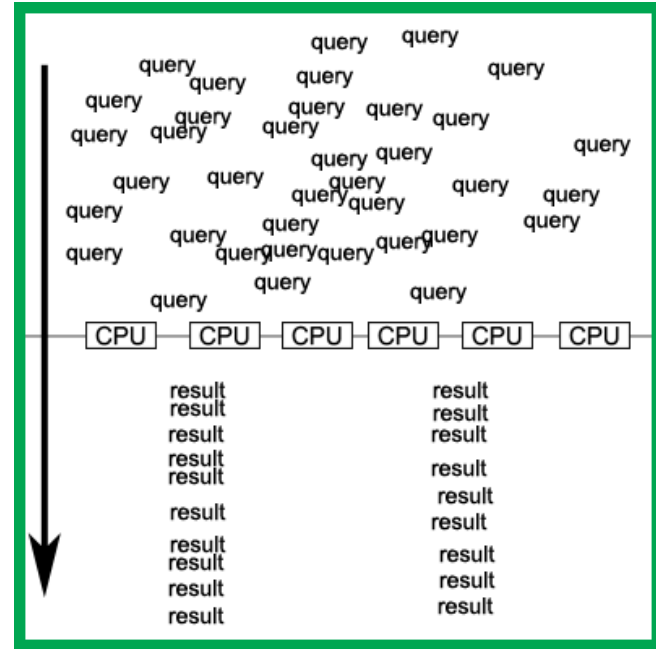
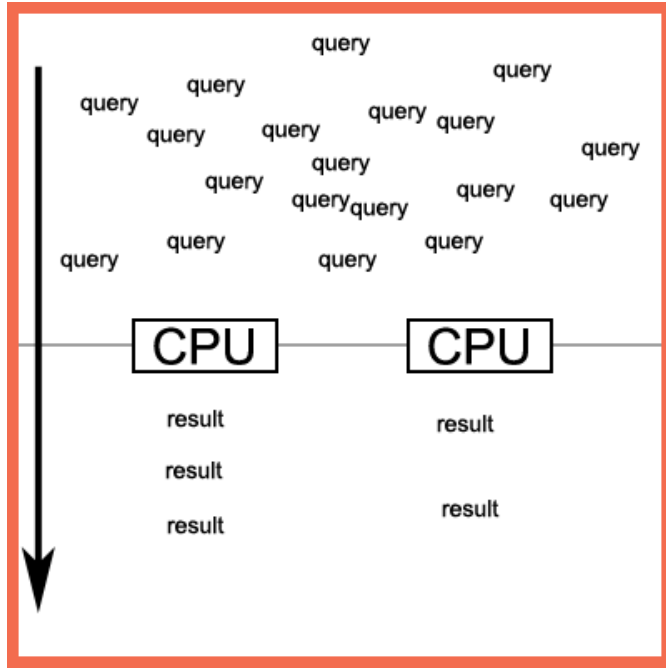
- Why do some search queries seem to stall while others are much more responsive?
- Investigation reveals the AWS EC2 free-tier imposes CPU throttling.



# Challenge 3: CPU Bound

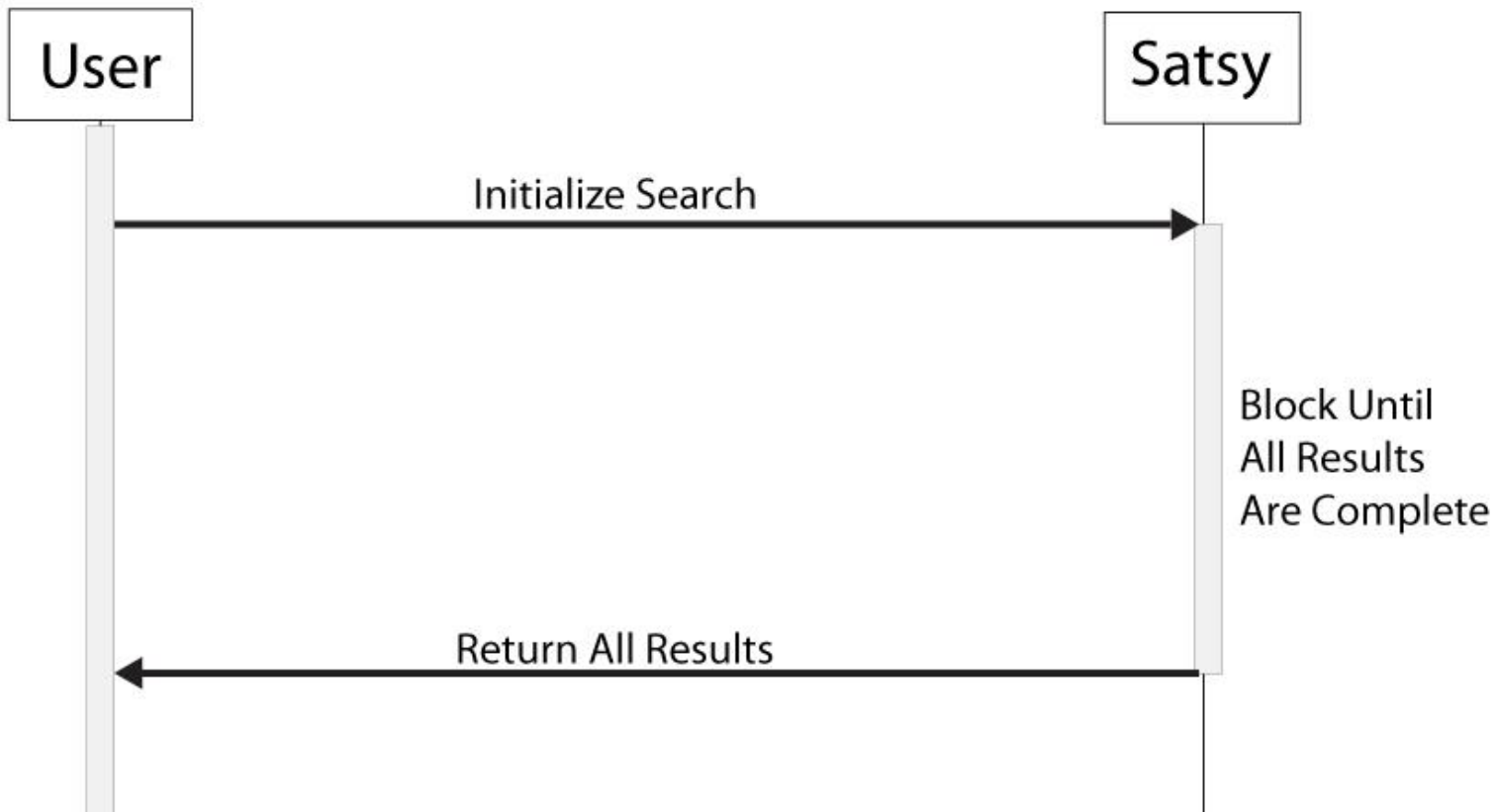
May 14  
13

- One Solver takes at least 60 Milliseconds of CPU time
- Searching 1000 database entries takes at least 1 minute



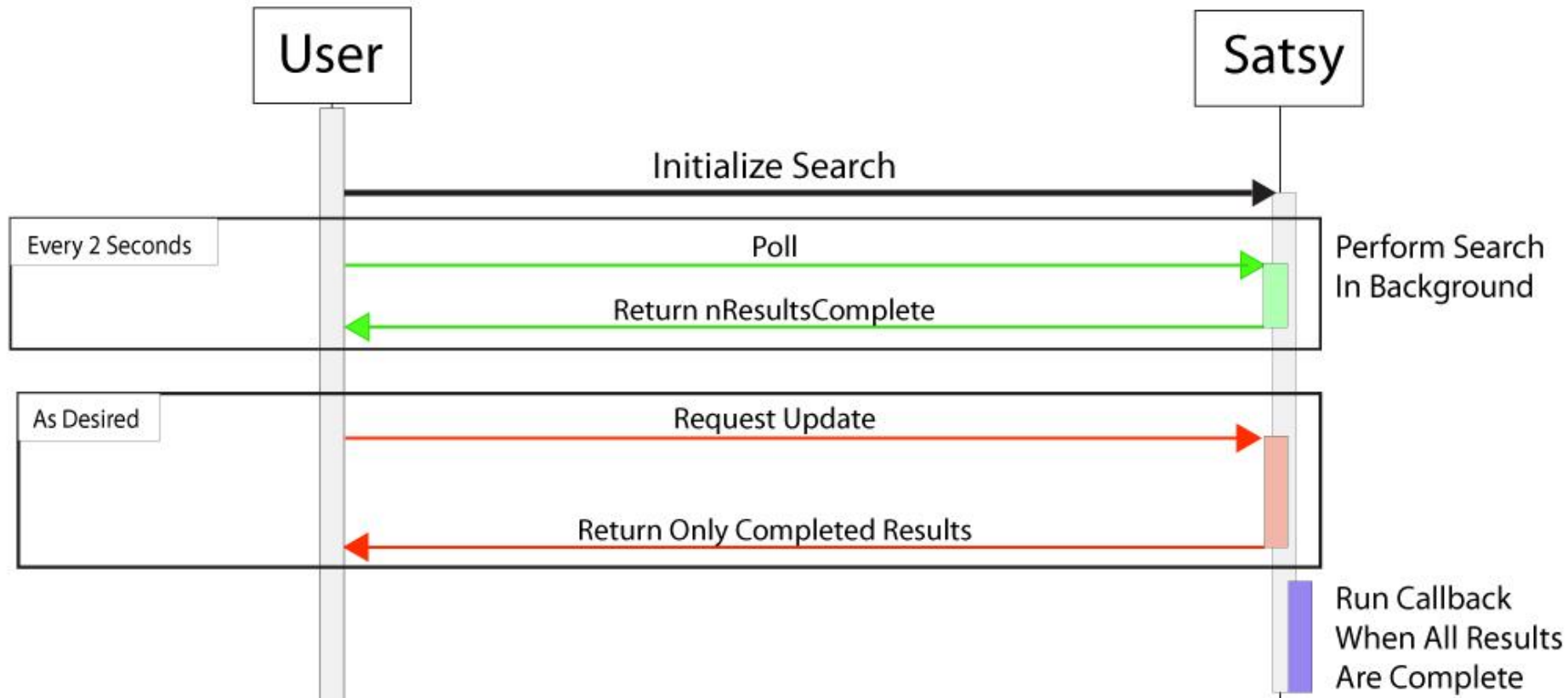
# Original Design

May14  
13



# Improved Design

May14  
13



# Initialize Search

May14  
13

1. end old search
2. prepare new ResultMap and Executor
3. create collection of Solvers
4. launch all the Solvers using the new Executor

# Solver Algorithm

May14  
13

1. check for search timeout
2. create SMT file
3. execute z3 and read result
4. increment result matrix
5. notify SearchCallback

# Result Matrix

May14  
13

	IO Pair 1	IO Pair 2
INITIAL	0	0
UNKNOWN	0	0
SAT	0	0
UNSAT	0	0
TIMEOUT	0	0
ERROR	0	0

# Database Structure

May14  
13

Table: boolean\_enc

Table: string\_enc

Table: char\_enc

Table: int\_enc

id: int  
num\_boolean: int  
num\_string: int  
num\_char: int  
num\_int: int  
method\_id: int  
enc\_method: longtext

Table: boolean\_src

Table: string\_src

Table: char\_src


Table: int\_src

id: int  
src: longtext  
thumbs: int




# Paths Encoded as SMT

May 14  
13

id	src
566	

```
public static boolean isLeapYear(int year){  
  if (year % 4 == 0) {  
    if (year % 1000 == 0 && year % 400 != 0) {  
      return false;  
    }  
    return true;  
  }  
  return false;  
}
```

source code for the method isLeapYear

id	nb	nc	ni	ns	m_id	enc_method
679	0	0	1	0	566	
680	0	0	1	0	566	

```
...  
(assert (= additive6106 (mod year int6105)))  
(assert (= boolean6115 true))  
(assert (= relationalEq6113 true))  
(assert (= int6100 4))  
(assert (= int6102 0))  
(assert (= int6107 0))  
(assert (= relationalEq6108 (= additive6106 int6107)))  
(assert (= relationalEq6108 true))  
(assert (= int6112 0))  
(assert (= int6110 400))  
(assert (= additive6111 (mod year int6110)))  
(assert (= relationalEq6103 true))  
(assert (= int6105 1000))  
(assert (= relationalEq6113 (= additive6111 int6112)))  
...
```

SMT code for an execution path through isLeapYear

# Ranking

- Generic superclass that can be extended to create new algorithms.
- Current algorithms are sorted by number of paths satisfied, percentage of paths satisfied and if all paths are satisfied.



# Front End Design

May14  
13

- Single Page Web App
- Asynchronous Communication
- Template-based presentation

523 -- 3

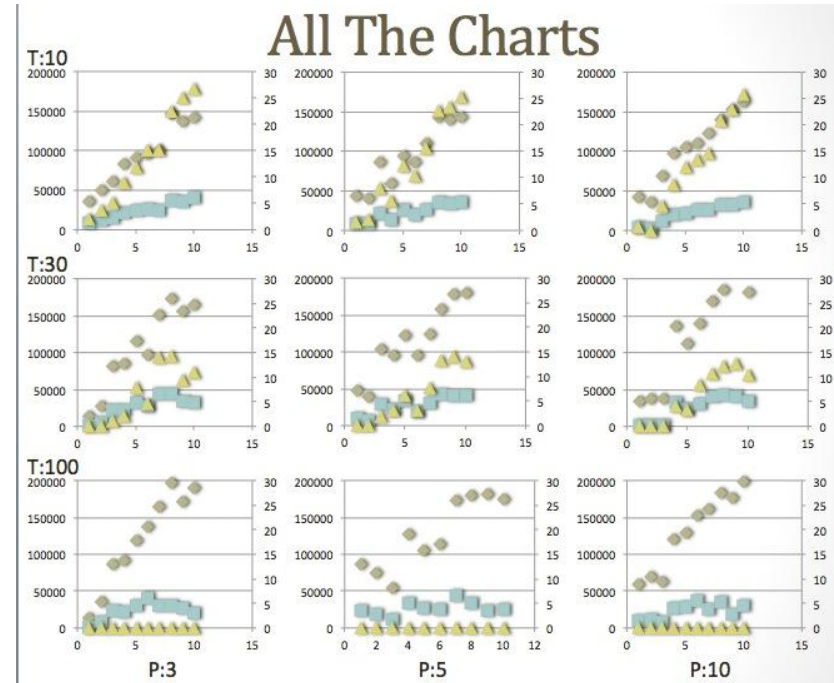
```
public static boolean isgraph(int ch){  
    return ('!' <= ch && ch <= '~');  
}
```

Show 22 More Results

# Testing 1: User Simulations

May 14  
13

- Strong Engineering Effort put into comprehensive tests
- Relied on Blocking/All-or-nothing nature
- Architectural change -> Out the window



# Testing 3: Revised Testing

May14  
13

- GUI: Manual “Mock” Flags
- Backend: Unit Tests
- Exceptional Cases: Database Integrity

# Testing 2: SearchCallback

May14  
13

- Shift to Asynchronous/Incremental DeliveryArchitecture
- Less control over timing, more “events”
- On completion call SearchCallback

# Questions

May 14  
13

