<table>
<thead>
<tr>
<th>Name</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sravan Bhamidipati</td>
<td>Tester</td>
</tr>
<tr>
<td>Michael Deeringer</td>
<td>Language Guru</td>
</tr>
<tr>
<td>Fang-Hsiang Su</td>
<td>System Architect</td>
</tr>
<tr>
<td>Jiacheng Yang</td>
<td>System Integrator</td>
</tr>
<tr>
<td>Chun-Yu Tsai</td>
<td>Project Manager</td>
</tr>
</tbody>
</table>

**Graph Modeling Language**

2013 Spring PLT

Team 4
Outline

Introduction: Why and How Gramola Language Highlights Project Management Gramola Translator Architecture Runtime Environment Test Plan Demo Conclusions
Why Gramola?

- **Graph Theory**
  - An important subject of modern science
  - Applied in numerous domains: social networks

- **Many languages support Graph, but they**
  - are not developer-friendly: Longer learning curve
  - focus on limited functionalities like graph DB or draw

- **Developers need the power of**
  - Graph-Native
  - Intuitive
  - Easy
  - Extensible
Language Highlights

Indentation-based blocks, logical lines

def Movie find_movie(str title):
    list<Node> setn = self.get_all_nodes()
    for Node n in setn:
        if isinstance(n, "Movie"):
            Movie movie = (Movie) n
            if movie.title == title:
                return movie
    return null
def void main():
    for object j in [2, 3]:
        int i = (int) j
        if i <= 3 and i > 2:
            print "i is less than 3"
        elif i + 1 <= 3:
            print "i + 1 less than 3"
class Actor(Node):
    str name
    def __init__(dict<str,str> dd, str actorname):
        Node(dd)
        self.name = actorname
str token1 = "login_token"
Graph fb1 = get_fb(token1)
draw(fb1, "name", "type")
dump(fb1, "PLT")
Project Management

• Weekly Scheduled Meetings
• Google Drive
  o Document management
• Googlegroups
  o Announcement
  o Meeting Agenda
  o Coordinating remote work
• GitHub
  o Gramola version control
Project Management

- Iterative and incremental project planning
- Project timeline
  - Gantt Chart
Project Management

- GitHub commits by days
Development Environment

- Frontend: PLY/Python
- Backend: Java
- Version Control: Git with Github hosting
- gcompile
Gramola Translator

Diagram:

- .gr
- Lexer
- Parser
- Analyzer
- Code Generator
- Java Class Maker
- Runtime Env.
  - Gramola Backend Lib
    - DB
    - Visual
    - Social
  - Tokens
    - NodeSet
    - Graph
    - Edge
    - Node
    - And more...
    - EdgeSet
    - Abstract
    - Syntax
    - Tree
- .java
- .class
Runtime Environment

- Java-powered runtime

```
gramola
- abstractdata
  - GraphElement.java
- datastruct
  - Edge.java
  - EdgeSet.java
  - Graph.java
  - Node.java
  - NodeSet.java
- testmain
  - ArraysTest.java
  - CoFriend.java
  - DumpTest.java
  - FBTest.java
  - GraphApp.java
  - LoadTest.java
  - OverAllTest.java
  - PathTest.java
- util
  - FBManager.java
  - GInformer.java
  - GraphDBController.java
  - GraphUtil.java
  - GraphVisualizer.java
  - LikePage.java
```

```
 INSTALL
```

```
.JVM
- gramola.jar
- third-party libs
```

```
gcompile
```

```
gexe
```

```
.Runtime Environment
```

```
gr
```

```
.class
```

```
Runtime Environment
```

```
COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK
```
Runtime Environment

• Scripts
  o INSTALL: one-click configuration of the runtime environment. For internal use and first-time user installation
  o gcompile: compile .gr -> .java -> .class
    ▪ ./gcompile hello.gr Hello
  o gexe: invoke JVM to link compiled user program (.class), the gramola library (gramola.jar) and other supporting libraries
    ▪ ./gexe Hello
Runtime Environment

• The Gramola library
  o Built-in data structures for graphs, e.g. Graph, Node, Edge, etc
  o Implementations of syntax sugars, e.g. initialize dictionaries with arbitrary number of key-value pairs
  o Converters/Drivers to connect to third-party libraries for advanced features, e.g. graph persistence, graph visualization
Test Plan

Pylint

• Fuzz testing: Lexer
• End-to-end automated testing: To localize bugs to a specific compiler phase.
• About 900 LOC of syntactically and semantically valid Gramola programs to test every keyword, operator, built-in, data type, data structure, programming construct through every phase of the compiler.
• Less focus on error-handling.
# Test Results

<table>
<thead>
<tr>
<th>Phase</th>
<th>Passed</th>
<th>Failed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lexical Analysis</td>
<td>35</td>
<td>0</td>
</tr>
<tr>
<td>Syntax &amp; Semantic Analysis</td>
<td>31</td>
<td>4</td>
</tr>
<tr>
<td>Code Generation</td>
<td>31</td>
<td>4</td>
</tr>
<tr>
<td>Compilation</td>
<td>30</td>
<td>5</td>
</tr>
<tr>
<td>Execution</td>
<td>30</td>
<td>5</td>
</tr>
</tbody>
</table>
Demo

1. IMDB
   a. Inheritance
   b. Loop
   c. Control Flow

2. Common Friends on Facebook
   a. User-defined class
   b. get_fb: Real-time data retrieval from Facebook
   c. dump: Graph data storage in Graph DB
   d. draw: Graph object=>dot => Graph visualization

3. More time?
   a. get_shortest_path: Shortest path finding
   b. Actually we have about 10+ Gramola apps!!
Conclusions

• We're proud of...
  o Gramola graph-native features (e.g., connection to FB), extensibility (class inheritance)

• What worked well...
  o gcompile
  o git version control

• Lessons we learnt
  o Start testing immediately after feature implemented!
  o We should plan for suitable scoping at the beginning!
We Can Do Better!

- Specific Features
- Attainable Target
- Yes, time is never enough
while work_in_same_place(team4):
    productivity[team4] += 1
Thanks!

© 2013 Spring PLT, Team 4: Gramola All rights reserved.

Title logo taken from http://www.estringita.com/pequeneces/2010/05/05/gramola/
RCA logo redux taken from http://kayleighmahon.wordpress.com/2012/09/