EMU: Element Manipulation Language

Philip Liou
Project Manager

Steve Pappas
Language Guru

Michael Wojcieszek
System Architect

Celia Hsu
System Integrator

Yi-Chen Lin
Verification & Validation

5/16/2013
Outline

Motivation
Introduction
Syntax
Translator Architecture
Software Tools
Run-time Environment
Compiler-Generator Tools
Test Plan
Project Management
Conclusions / Demo
Why EMU?
Motivation

Why Emu?

• Easy tasks with easy tools
Motivation

Why Emu?

- Easy tasks with easy tools

```java
type Song;
type Artist;
$Artist art1 = "Kelly Clarkson".add();
$Song kc1 = "Stronger".add();
$Song kc2 = "Dark Side".add();
$Edge e = <kc1, kc2, 3.0>.add();
art1 -> [kc1, kc2];
```
Introduction

Shan-Chi Hsu (System Integrator)
• **EMU** is a programming language that helps the expression and manipulation of relationships between data by graph-like structures.

• **Node** represents an object with a specific type defined by developers.

• **EMU** provides two ways of expressing relations between *Nodes*: *references* and *edges*.
Introduction

• What are the characteristics of EMU?
  • **Intuitive**: The structure is similar to a general graph.
  • **Familiar**: Much of the syntax is similar to that in Java.
  • **Visual**: A built-in function is provided to visualize data.
  • **Reusable**: Object construction code can be used across multiple programs.
  • **Flexible**: The construction is not tied to a rigid schema.
Syntax

Shan-Chi Hsu (System Integrator)
Syntax

- **Data Types**: int, double, string, array, Edge, Node
- **Control Statements**: if/else, while, for, foreach
- **Basic operators**: =, ==, !, &, |, >, <, >=, <=, +, -, *, /, %, +=, -=, *=, /=, %=, ++, --, [], ()
- **String operators**: +, +=
- **Node/Edge operators**: <>, .refs(), .edges(), .value, .type, ->, <->, *
- **Array operators**: .length, +=, -=
- **Built-in functions**: print(), graph(), .opposite(n), .add(), .remove()
# Declare Node types
type Song;
type Artist;
# Create an Artist node with value “Kelly Clarkson”
$Artist art1 = <"Kelly Clarkson">.add();
# Create Song nodes
$Song kc1 = <"Stronger">.add();
$Song kc2 = <"Dark Side">.add();
# Create an edge between kc1 and kc2 with weight 3.0
$Edge e = <kc1, kc2, 3.0>.add();
# Create references of the Artist node to Song nodes
art1 -> [kc1, kc2];
# An equivalent way to create references
# art1.refs() += [kc1];
# art1.refs() += [kc2];
# Declare Node types
type Song;
type Artist;
# Create an Artist node with value “Kelly Clarkson”
$Artist art1 = <"Kelly Clarkson">.add();
# Create Song nodes
$Song kc1 = <"Stronger">.add();
$Song kc2 = <"Dark Side">.add();
# Create an edge between kc1 and kc2 with weight 3.0
$Edge e = <kc1, kc2, 3.0>.add();
# Create references of the Artist node to Song nodes
art1 -> [kc1, kc2];
# An equivalent way to create references
# art1.refs() += [kc1];
# art1.refs() += [kc2];
Translator Architecture

Mike Wojcieszek (System Architect)
Translator Architecture

Source Program

1. Scanner (emu.flex)
2. Parser (emu.y)
3. Semantic Analyzer

Code Generator

Java Library

Jung Library

EMU Compiler

JAVA Compiler
Translator Architecture

Source Program

Scanner (emu.flex)

Parser (emu.y)

Semantic Analyzer

Code Generator

# hello.emu

$node n = <"Hello, EMU">.add();

print(n.value + ";

)graph();

Java Library

Jung Library

Egg

Symbol Table

.java

.class
Translator Architecture

ProgramItems:
  ProgramItem:
    Decl:
      Initializer:
        TypeSpec:
          EMUTYPESPECIFIER: $Node
        AssignmentExp: =
          IDNode: n
       InitializerExp:
          STRINGLITNode: "Hello, EMU"
        MemberCall: add

ProgramItems:
  ProgramItem:
    Stmt:
      ExpressionStmt:
        Expression:
          FunctionCall: print
            Expression:
              AdditiveExp: +
                PostfixExp: dot
                IDNode: n
            MemberCall: value
            STRINGLITNode: "\n"

ProgramItems:
  ProgramItem:
    Stmt:
      ExpressionStmt:
        Expression:
          FunctionCall: graph
/* hello.java - auto generated by emuc at 2013-05-15 19:30:40.892 */

import emulib.core.*;
import java.io.*;
import java.util.*;

public class hello {

    private static Printer _pEMU = new Printer();
    private static Graph _gEMU = new Graph();

    public static void main(String args[]) throws IOException {
        Node n = new Node("Node", "Hello, EMU");
        _gEMU.add(n);
        _pEMU.print(n.getValue()+"\n");
        _gEMU.visualize();
    }
}
Software Tools

Steve Pappas (Language Guru)
Software Tools

- Sublime Text
- Eclipse
- IntelliJ IDEA
- Java
- Perl
Software Tools - Scripts

**build** - Makefile

- ./build
- ./build -t <test_file>
- ./build -c

**emuc** - Compiler Script

- ./emuc source.emu
- ./emuc -x source.emu

Flags:
- h or -help: Display help text
- c or -clean: Remove build and temporary files from directory
- t or -test: Run all regression tests and determine PASS or FAIL
- p or -print: Used in conjunction with -t, just print AST for each regression test
- g or -generate: Used in conjunction with -t, just generate solution file for specific test case
Run-time Environment

Steve Pappas (Language Guru)
Run-time Environment
Compiler-Generator Tools

Yi-Chen Lin (Testing and Validation)
Compiler-Generator Tools

**Lexer**

**Parser**

**Visualization Library**
Test Plan

Yi-Chen Lin (Testing and Validation)
Test Plan

Iterative

• Bug-fixing
• Golden samples generating

![Diagram showing Test Plan]

- Target Sample
- Module
- Golden Samples
- Expected Output
- Unexpected Output
- Error Feedback
Test Plan

Regression

• component adding
Project Management

Philip Liou (Project Manager)
Project Planning

- Define Roles and Responsibilities
- Set Milestones

Project Management

- Task Tracking
- Frequent Progress Emails

Collaboration and Communication

- Suite of Collaboration Tools
- Meeting Rooms
Demo
Conclusions

Yi-Chen Lin (Testing and Validation)
Conclusions

Why EMU?
• Easy to use for visualizing relationships

What could be different?
• Tools research
• Time management

Lesson Learned
• Gauge the expectation
• Communicate with others
• Meet with mentors
• Don’t fall behind
• Use line numbers
Thank you!