Virtual Machine using Object-Oriented Computing

Jared Rainwater[1], Steve E. Watkins[2], and Benjamin Cooper[3]

[1]Computer Science Department and
[2]Electrical and Computer Engineering Department, Missouri University of Science and Technology, Rolla, Missouri USA and
[3]Simplex Technologies, Rolla, Missouri USA
jqrvz6@mst.edu steve.e.watkins@ieee.org, ben.cooper.seed@gmail.com

Abstract

A design for computing systems that is based on object-oriented computing is investigated. The approach, which is called Seed since it is patterned after biological systems, consists of Seed virtual machine (VM), Seed composer, and Seed user interface (UI). The VM facilitates all the computations happening in the system. The composer breaks up the code to be ran into smaller segments for future optimization and more fine-grained control over what is running. The UI is the interface a user will interact to use the system. This project examined the feasibility of an object-oriented VM based on C++ programming. The VM was implemented with an object-oriented memory system to store all the data being computed on, an interpreter which executes the bytecodes associated with computation and updates the data in the Object Memory as need, and finally, a controller which creates test objects in the object memory system. The Seed VM allows for enhanced user control and optimization and has the potential for run-time reduction. By contrast, the traditional approach implements the operating system and application functions as distinct, monolithic code. In addition to implementing a working Seed VM, run-time benchmarks were identified for future performance testing. Other future work includes implementing the composer and the UI and testing the integrated Seed system. This preliminary project shows a basic, object-oriented VM implementation and highlights advantages of optimized coding segments and fine-grained user control. Compatibility between Seed systems and traditional systems in a hybrid environment is a concern.

Keywords

Object-Oriented Computing, Programming, Software.

Virtual Machine Using Object-Oriented Computing

Jared Rainwater, Steve E. Watkins, and Benjamin Cooper

Author Affiliations

J. Rainwater - CpSc Department
S. E. Watkins - ECE Department
Missouri S&T, Rolla, MO
B. Cooper - Simplex Technologies
Rolla, MO

Project:
Implement a computing system using object-oriented computing design
"Seed" Project:
Name emphasizes the bio-inspiration for the object-oriented approach

Project Overview

Overall demonstration project for an entire computing system with the following elements

- Seed VM: Virtual Machine for all computations
- Seed Composer: Segments computing objects
- Seed UI: User Interface

VM is the subject of this work.

VM Functions

- Object memory to store data
- Interpreter to execute bytecodes
- Controller interface between the interpreter and the rest of the VM

VM implemented in C++

Comparison of Approaches

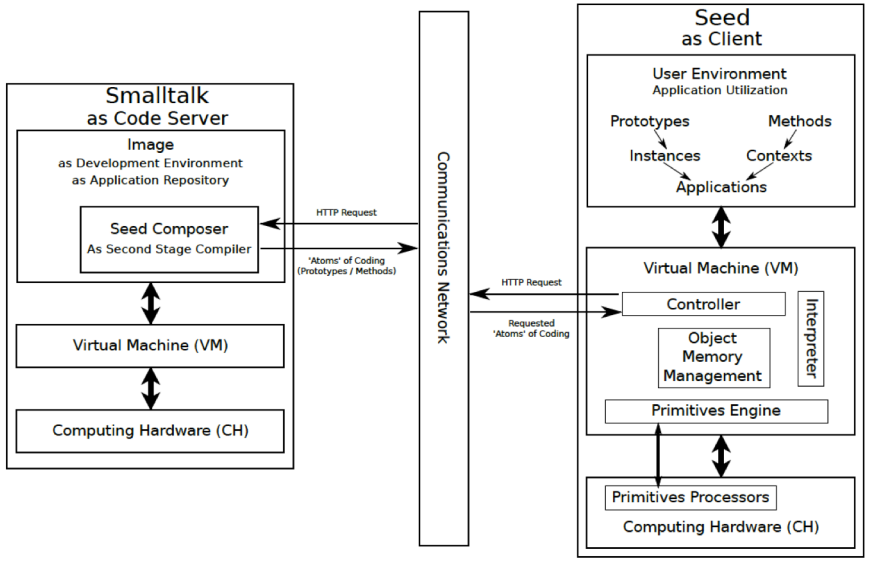
Object-Oriented Approach

- Able to split code up into segments with goals to improve efficiency, flexibility, and security
- Operation can be lively like SmallTalk across the entire system. (Updates are instantly broadcasted across all devices)
- Users have more visibility of the code running on their system

Traditional Procedural Approach

- Code is monolithic with one executable or several large executables
- Updating code requires compiling or re-running the entire code base
- Users require advanced knowledge on reverse engineering to know what is running on their system
- It is a proven method in implementation

Plan for Expanded Computing Environment



VM Accomplishments

- Object Memory: Fully functional
- Interpreter: Executes arithmetic and control flow bytecodes
- Controller: Puts test objects into memory and performs arithmetic operations on them

Implementing the full Seed system is ongoing work.

Next Steps

Challenges to integrate the Object-Oriented VM with other systems

- Hardware compatibility
- OS compatibility
- General compatibility in hybrid environment.

Lessons Learned

- Fundamental differences exist among design approaches
- Programming platform is important, e.g. SmallTalk may be a better choice for this approach
- Run-time may be a useful metric for comparing approaches
- Compatibility with traditional systems is a concern in hybrid applications

Contact: ben.cooper.seed@gmail.com