Fifi: An Architecture to Realize Self-evolving of Java Program

Ming-Yang Hou
ehmy@263.net

Xi-Yang Liu
xyliu@xidian.edu.cn

He-Hui Liu
hhliu@mail.xidian.edu.cn

Software Engineering Institute

Xidian University, China

ICSE 2006
Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)
22 May 2006, Shanghai, China
• **Code need to modify themselves while running**
  - All depends on human: test, bug fix, deployment…
  - Not make the most of computation resource and knowledge

• **If code can modify themselves while running**
  - Keep learning, testing and evolving by itself
  - Without the participation of mankind
  - Increasing intelligence
  - Support time critical mission and long-lived application – hot fix
  - Better support for Adaptive and Self-Managing Systems

• **We name this as 'Self-evolving'**
  - Expansion to Self-*

Motivation

Software Engineering Institute, Xidian University, China
• Approach for Self-evolving in JAVA
  • How to realize Self-evolving in JAVA
• Fifi Self-evolving architecture
  • Propose a new architecture: Fifi
• Applications of Self-evolving
  • Where to apply Self-evolving
• Further researches
  • What to research next
Steps of Self-evolving

• **Runtime monitor**
  - Watches the execution of itself
  - Identifies the need of evolution

• **Static bytecode evolution**
  - Analyze static and runtime information of itself
  - Modifies code of itself to satisfy the evolution requirement

• **Dynamic code switch**
  - Switch old version code to new version dynamically
  - Release old version code

1. Find his shortage
2. Tailor for himself
3. Then, live a running life

Software Engineering Institute, Xidian University, China
Supports from JAVA to Self-evolving

- **Java Platform Debugger Architecture (JPDA)**
  - Java Virtual Machine Tool Interface (JVMTI)
  - Java debug interface (JDI)
  - Use for monitor and control

- **Bytecode**
  - Self-describe and easy to analyze and instrument
  - Use for evolution

- **Java virtual machine**
  - Replace old code to new code dynamically
  - Use for dynamic code switch
We need an architecture

• It should support Self-evolving in a structural way
  • General purpose
  • Stable, secure, efficient

• If 'everything is computing'
  • Everything has a computational mode
  • Learn architecture from reality

• Learn from nature life: *Human*
  • Evolved for a very long time
  • On the top lever of biology evolution
    • Head, body, extremity
    • genetic system, nervous system, endocrine system, immune system
• **Master VM (MV)**
  - Contains Monitor, Evolver, Agent
  - Monitor
    - Spies and control the running of Target VM
    - Sends result to Evolver
  - Evolver
    - Makes the decision whether and how to evolve
  - Agent
    - Communicate with Target VM
    - Directs Monitor to acquire information.

• **Target VM (TV)**
  - Function units (Planner, Executer, Supervisor)
  - Planner
    - Makes execution plan for Executer.
  - Executer
    - Carry out execution plan
  - Supervisor
    - Monitors the execution of Executer
    - Gives feedback to Planner
Features of Fifi

- **Two parallel virtual machines**
  - Do not interfere with each other
  - keep TV as simple as possible
  - MV can attach to another TV
  - MV can run in remote

- **Three loop backs**
  - **Autonomy loop back**
    - Made up of Monitor, JVMTI, Function Unit, Agent
    - Dual bidirectional channel for collecting information and control running
    - Can be used for self-control
  - **Evolving loop back**
    - Made up of JVMTI, Monitor, Evolver
    - Drive Evolver keep evolving
  - **Homeostasis loop back**
    - Made up of Planner, Executer and Supervisor in each function units
    - Can lead function units into a controllable and stable state

Software Engineering Institute, Xidian University, China
Viewpoints from bionics

Component Equivalence
- Master VM ⟷ Head
- Target VM ⟷ Body
- Function unit ⟷ \{Organ, Cell\}
- Code ⟷ DNA

Loop Back Equivalence
- Autonomy loop back ⟷ Nervous system
- Homeostasis loop back ⟷ Endocrine system
- Evolving loop back ⟷ \{Genetic system, Immune system\}

Fifi ➔ Electronic life ➔ \{Autonomy, Homeostasis, Self-evolving, Adaption\}

Software Engineering Institute, Xidian University, China
Implementation of Fifi

- Virtual Machine
  - Java2, Hotspot Virtual Machine

- Project build
  - Eclipse 3.1.1

- Monitor
  - Based on package com.sun.tools.example.debug.TTY in JDI reference implementation from SUN
  - Commands were wrapped as atom action

- Evolver
  - Based on open source project ASM2 from ObjectWeb use Visitor design pattern
  - Specific evolution knowledge will be added
  - CodeIndex of current bytecode position, stack position of next local variable has been added
## Prototype selection for Fifi validation

### Goal

**Catch 'Divide by zero' error automatically**

### Exception table:

<table>
<thead>
<tr>
<th>from</th>
<th>to</th>
<th>target</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>6</td>
<td>9</td>
<td>java/lang/Exception</td>
</tr>
</tbody>
</table>

### Before evolution

```java
public int Cal(int a, int b) {
    int c=0;
    c = a / b;
    return c;
}
```

### After evolution

```java
public int Cal(int a, int b) {
    int c = 0;
    try {
        c = a / b;
    } catch (Exception e) {
    }
    return c;
}
```

### Before evolution (Code)

```java
public int Cal(int a, int b) {
    int c=0;
    c = a / b;
    return c;
}
```

### After evolution (Code)

```java
public int Cal(int a, int b) {
    int c=0;
    c = a / b;
    return c;
}
```

### Before evolution (Machine Code)

```
0: iconst_0
1: istore_3
2: iload_1
3: iload_2
4: idiv
5: istore_3
6: iload_3
7: ireturn
```

### After evolution (Machine Code)

```
0: iconst_0
1: istore_3
2: iload_1
3: iload_2
4: idiv
5: istore_3
6: goto 11
9: astore 4
11: iload_3
12: ireturn
```
Prototype design

Monitor watches the running

Suspend all thread after error occur

Collect information about this error

Is it a 'Divide by zero' error?

Locates the instructions where error occur

Evolver Add Try-Catch clause

Commits evolved class to Target VM

Pop out current frame (method)

Continue running (enter again)

- Many error were embedded
  - At different position
  - In different method
  - In different class

- Extended test
  - Simulate classes interaction
  - Methods overload
Prototype testing

• Test result
  • Runs successful and peaceful to the end
  • Each 'Divide by zero' error is caught and evolved automatically

• Conclusion
  • Continuous evolution on Fifi has been achieved
  • Fifi has been partly validated
  • Need more prototype: re-implementation of AI algorithm

• Improvements to be done
  • Virtual Machine
    • Class remote load and query
    • Hot switch directly
    • No binary compatible restriction
  • JVMTI
    • Random jump within a method
    • Automatically rollback
    • Code instrumentation
    • Evolve source code
Blueprint of Self-evolving

Master VM

Target VM

Species 1

Species 2

Species 3

Software Engineering Institute, Xidian University, China

FiFi Self-evolving Architecture
• Adaptive and self-managing
  • Sense the environment, inspect the running of itself
  • Control and modify itself, find and fix runtime error or fault automatically

• Machine learning
  • Create code according to learned information
  • Learned result can learn and action initiatively and continually

• Artificial intelligence
  • Rearrange code arrangement dynamically

• Software maintenance
  • Record and analyze runtime information automatically
  • Bug hot fix and dynamic upgrade, new component hot integration

• Software test
  • Alter code and variable, change and control program running routing

Software Engineering Institute, Xidian University, China
Further research issues

- Environment support Self-evolving
  - Simulation of the nature, a virtual world

- Evolving and variation engine
  - Species evolve to different directions while competes with each other
  - Base on pre-learned knowledge or makes random variation choice

- Rulers for wash out
  - Acts the same as the ‘survival of the fittest’ in the nature
  - Every species must obey

- Code self-description and analysis
  - Language more suitable for self-evolving, analysis and modification
  - Based on static and runtime information

- Self-evolving of above domains
Summary

• **Approach for Self-evolving in JAVA**
  - Runtime monitor - JVMTI / JDI
  - Static bytecode evolution - Bytecode instrumentation
  - Dynamic code switch - Virtual Machine

• **Self-evolving architecture: Fifi**
  - Master VM, Target VM
  - Three loop backs
  - **Success in continual 'Divide by zero' error evolution**
  - **Partly validated Fifi Self-evolving architecture**

• **Applications of Self-evolving**

• **Further researches**
  - Electronic life lives in virtual world
Thank you!

Question?