**CSFV**

**University of Washington**

**Monthly Technical Report for November, 2012**

Reporting period: 1 November 2012 – 30 November 2012

Date of Report: 15 December 2012

Project Title: Verigames

Contract Number: FA8750-12-C-0174

Program Manager: Dr. Drew Dean, DARPA I2O

Submitted by:

Michael Ernst / Zoran Popović

Computer Science & Engineering

University of Washington

AC101 Paul G. Allen Center, Box 352350

185 Stevens Way

Seattle WA 98195-2350

Fax: (206) 543-2969

E-mail: {mernst, zoran}@cs.washington.edu

**Distribution List**

DARPA/I2O

ATTN: Drew Dean

team-dean@darpa.mil

3701 North Fairfax Drive

Arlington VA 22203-1714

AFRL/RITA

ATTN: Dilia Rodriguez

Dilia.Rodriguez@rl.af.mil

525 Brooks Road

Rome NY 13441-4505

**Unclassified - For Official Use Only**

# Verigames - Project Progress

**1. Expected Progress This Month**

Support the new class file format in Java 8.

Finish the TestSolver.

Finish up divide by zero type system case study, and convert the type system into a game.

Finish extending the Annotation File Utilities to allow insertion of casts in arbitrary locations.

Begin integrating the new dataflow framework into Verigames.

Brainstorm additional visual metaphors for the game (besides pipes and traffic) that we can test for engagement.

Continue the hiring process.

**2. Accomplishments This Month**

The TestSolver is complete. It is comprised of two test types: Constraint Tests and XML Tests. Constraint tests are used to verify that a Visitor and Checker are creating the correct constraint information as they traverse through source code. Each test runs an inference, prints the generated constraints, and compares them to a file of expected constraints. Constraint tests use the first half of the source-to-game pipeline. XML tests verify that constraints are correctly converted into an XML representation of a game. Each test runs an inference and compares the generated XML to a file of expected XML. Because layout information may change slightly from run to run, layout is removed from the XML files before the comparison. XML tests use the entire source-to-game pipeline.

Support for the new class file format in Java 8 is well underway – Oracle has made a preliminary build of javac that incorporates our changes publicly available. We anticipate completing this in the next month.

We implemented the visual representation of map keys and other secondary type systems (something we’ve referred to internally as “stars on cars,” owing to how we are currently representing them). Thinking through these properties now will help us ensure that we have a generic visualization system for the game that can accommodate all of our desired type systems. Taking the Nullness Checker as a concrete example, there are three distinct properties that need to be represented:

* nullable vs. non-null (road/pipe width -- only 2 possibilities)
* map keys (colored stars -- many possibilities, including no color)
* initialized vs under-construction (2 possibilities)

Most other type systems will be simpler than this one, and some of them will just have two possibilities to represent. Thus, if we can handle the Nullness Checker well, then we are likely to be able to handle most of the type systems.

We designed the visualization for connecting method arguments to methods. Two models were proposed: in the first, we pass arguments straight through the sub board. In the second, we split, and only connect one of the resulting pipes to the sub board. Additionally, there are no outgoing pipes for arguments. The first has the advantage that it is visually cleaner, and semantically it allows a method to change the type of an argument, which could conceivably be useful after flow is implemented, because a method could change some property of an argument (maybe adding it to a map, so it becomes a key for the map). This would be represented by a pipe that is different above than below. The second model explicitly prevents methods from changing the types of the arguments. We decided to go with the first method, as it is cleaner may be more compatible with flow sensitivity.

We performed work on the model-view-controller in order to cleanly separate the game visualization from the game logic. This allows us to easily switch out different visual metaphors in response to user testing and potentially user preference. We are exploring a skin that uses the metaphor of pneumatic tube messages in a large building. This metaphor has the advantage of feeling more concrete than the totally abstract pipes and balls, while avoiding situations where the “traffic jam” theme placed roads and cars in situations that could seem strange and nonsensical.

We began work on framework for the tutorial section of the game. The tutorial section will be an important part of teaching players how to play the game and will be required before any semi-formal or formal playtesting can begin.

We continued to coordinate with TopCoder on game site design. We had two main suggestions on the final proposed wireframe. The first is a "groups" feature, as we believe the gameplay boards generated from large programs may lend themselves to teamwork of this kind. The second suggestion was for multiple leaderboards as opposed to one single leaderboard, with the idea that players having multiple different vectors to compete on would encourage a greater number of players to feel good about their individual contribution.

We hired one additional software engineer, Suzanne Millstein, who will start on January 7. We continued candidate search, with over a dozen interviews.

**3. Deliverables Submitted**

N/A

**4. Publications Made**

We submitted papers related to pluggable type-checking and our analysis framework to PLDI 2013 and ECOOP 2013.

**5. Meetings**

* Weekly UW Verigames team meeting
* Weekly integrator conference call

**6. Issues or Concerns**

Iterative game development cannot start until our IRB is approved. Our game design and development process is highly reliant on gathering data from real players and making adjustments to the game based on perceived patterns of play. Our inability to collect and study data in the short term means that there are a number of game design questions that will remain open for the time being.

**7. Plans for Next Month:**

* Support the new class file format in Java 8.
* Get cast/receiver insertions with multiple annotations working. Get inner type annotations with casts/receivers working.
* Continue integrating the new dataflow framework into Verigames.
* Introduce the idea of game “sessions” into the game in order to handle multiple levels served from the integrator.
* Begin thinking about how the game handles (visualizes) very large input programs.
* Begin creating art for the next visual metaphor for the game.
* Create a “first pass” tutorial sequence with simple programs in order to introduce players to the concepts of the game.
* Continue hiring.

**8. Financial Summary**

November: Projected expenditures for the month were estimated at $110k. Actual expenditures were $66k. The first invoice from Julia Srl has been received and will post in December. Julia Srl has hired two staff who will begin in January. We hired another software engineer (Suzanne Millstein), who will start working on January 7.