Weekly Report

Team Dec 15-12: PaniniJ Eclipse Plugin

Week 4: *Feb. 9th - Feb. 16th*

Advisor	Dr. Rajan	
Client	Dr. Rajan	
Team Members	Dalton Mills	*Webmaster*
	David Johnston	*Team Lead*
	Kristin Clemens	*Communication Lead*
	Trey Erenberger	*Key Concept Holder*

Weekly Summary

Interoperability, A New Project Objective: During the Friday meeting with Dr. Rajan, we were given a new project objective: make the capsule code which we are generating interoperable with capsules generated by the existing PaniniJ compiler, `panc`. By this, we hope to make it so that the `.class` files generated by `@PaniniJ` are usable in a `panc` project and vice-versa.

Duck Futures (a.k.a. Transparent Futures): Discussions with Dr. Steve Kautz and Dr. Rajan helped give us some very useful background into an efficient existing mechanism for procedure invocation in `panc`-generated capsules: *duck futures*. The basic idea behind them is that the capsule client does not know that the value immediately returned from a procedure invocation is actually a future. Duck futures can be used as an additional optimization to reduce the number of objects created during a procedure invocation: the duck future immediately returned to the client doubles as the callable/runnable object which encloses the arguments and is placed onto the capsule's FIFO message queue.

Project Refactoring: See the "Technical Progress" and "Tentative Plans" sections.

Technical Progress

This week we have made a few minor technical changes to our existing proof-of-concept work and some some research into additional technologies which we expect will be useful.

- The existing proof-of-concept capsule generation code was moved to `dwtj/panini` (which is a fork of `hridesh/panini`).
- With a little bit of direction from David, Trey revised and simplified the means by which a client constructs a capsule: they no longer use the constructor directly. Rather, a static factory method is used to build a capsule and start it running.
- Kristin has been investigating and researching Maven and how it might help simplify our both our development build process and the build process of clients using our deliverables.
- Dalton first spent some time working with Java 6's annotation processor API and has moved onto The Checker Framework's abstract type processor API. We expect that the latter API will potentially be the key to performing various type checks on
    - capsule template code,
    - capsule client code (i.e. code which uses auto-generated capsule classes), and
    - capsule designs.

Meetings

Weekly Administrative Meeting

Members Present: All
Additional Participants: N/A
Date & Location: Thursday 12 of February; Molecular Biology 1414
Minutes:

- Meeting Start: 1:10
- Discussion of explicit, testable requirements for project
- Discussion of features
    - what comes for free with this approach
        - basic java syntax, reporting space for warnings/errors
        - IDE/compiler independent
- Agenda overview for Rajan meeting on Friday
    - Walkthrough of panc generated java code
    - Comparison to prototype implementation
    - Presentation of feature list to Rajan
- Drafting of feature list
    - Capsule property checking
        - Incrementally add more of the formal validation currently implemented in panini
    - Support of different thread profiles
    - Polymorphism compliant
    - 1 object created per feature call
- Discussion of research findings from past week
    - Dalton: CheckerFramework exploration
        - CheckerFramework may rely on specialized javac until Java 9
        - CheckerFramework seemingly opaque (possibly too opaque?)
    - David:
        - Futures: Explicit vs Invisible
            - capsule class procedures will return the data type that the template dictates
            - this will require generated classes of the return type that extend the return type.
            - this will eliminate the need for the user to call and plan for futures and result in a cleaner, more readable code
- Work on Feature list to show to Dr. Rajan

Bi-Weekly Advisor Meeting

Members Present: Trey, David, Dalton
Additional Participants: Dr. Rajan
Date & Location: Friday 13 of February; Atanasoff 101
Minutes:

- Definition of ‘surpassing expectations’
- Coffee break
- Guided tour of generated java code from panc
    - naming convention break explanation
    - optimization techniques revealed
- Explanation of Panini duck typing
- New target to interoperate with panc-compiled capsules.

Weekly Collaboration Meeting

Date & Location: Sunday 15 of February; Google Hangouts
Members Present: All
Additional Participants:
Minutes:

- Progress report since meeting with Dr. Rajan
- Dr. Rajan meeting update for Kristin
- Discussion of impact of new developments on existing prototype
    - new namespace conventions
    - new repository
    - interoperability between generated code from panc
    - panini.runtime target
- Delegation of tasks for week 5
    - Move to panini repo provided by Dr.Rajan
    - Migrate to naming conventions used in panc
    - Get hello world type checker
    - Change code generator to panc convention

Individual Hourly Contributions

Trey Erenberger	8 Hours
David Johnston	8 Hours
Kristin Clemens	8 Hours
Dalton Mills	7 Hours

Culumulative Time Contribution

Trey Erenberger	27.5 Hours
David Johnston	42.5 Hours
Kristin Clemens	23.5 Hours
Dalton Mills	25 Hours

Tentative Plans for Week 5

- The existing code in `dwtj/panini` needs to be refactored in a number of ways.
    - Kristin is planning on setting up an improved Maven project. Ideally, this Maven project will include (1) panc runtime code as a dependency and (2) the previous work in `dwtj/capsule-generator-tests` into dwjt/panini in such a way that emulates the build process of a capsule client (i.e. basic integration tests).
    - Trey is planning on refactoring the existing work to fit into the packaging/naming scheme laid out by Dr. Rajan.
- Once Trey has done this basic refactoring, he and David are going to consider what parts of the code-generation should be revised next. Making this decision will likely require a better understanding of the existing `panc` runtime code and how we can make use of it.
- Dalton will be continuing his research into pluggable type checking with The Checker Framework. His immediate goal is to create an extremely simple “Hello, World.” style type checking example (e.g. test that a particular variable is set to some arbitrary number, say, five). Once that has been example has been completed, Dalton and David are going to work together to try to implement a very basic escape analysis type checker.