Practical Units Checking of Large Programs

Jian Xiang, John Knight, Kevin Sullivan

Department of Computer Science

University of Virginia

Charlottesville, VA USA

{Jian,Knight,Sullivan}@cs.virginia.edu

*Abstract*— Units checking

Keywords— Units checking, dimensional analysis, real-world types, software assurance

# Introduction

Units checking and dimensional analysis is used extensively in Physics to detect algebraic errors. Software computes using a digital representation of physical quantities, and the introduction of units checking and dimensional analysis into the analysis of software offers the potential of static detection of software faults.

In this paper we present a summary of a toolset that supports comprehensive units checking and dimensional analysis of large software systems together with an empirical study of the use of the toolset.

An example of a major failure for which one of the causal factors was mismatched units is the 1999 Mars Climate Orbiter. The spacecraft was lost because different parts of the software system used different units of measurement [].

Units checking is . Dimensional analysis is.

Several researchers have explored the possibility of introducing both units checking and dimensional analysis into software. To do so, a mechanism is required to associate real-world properties with items in the machine world. Typically this is done with annotations.

The results of previous research are a promising indication of the value of units checking and dimensional analysis. But they are limited because.

Limitations of prior work (very limited results and crude tools based on annotations). Solution objectives – tools and technology that work for large programs plus empirical study of large program(s).

# Logic Interpretation

Brief summary of the theory leading to explanation that units and dimensions are important special cases.

In separate work we have introduced the concept of the interpreted formalism and real world types.

# Toolset Overview

The toolset is cool. Provide overview and point to the tools paper.

Explain enough to be able to show that the tools meet the objectives for units analysis.

# Subject Systems

Explain that we didn’t write these systems. All we did was grab the code and analyze.

## Flight Planner Project

The interpreted formalism has been evaluated on two open-source software projects. Both projects are from the geography domain. The two projects are:

 The Kelpie flight planner [43]. This is an open-source Java project based on FlightGear [25]. The Planner project uses the airport and navaid databases of FlightGear to determine routes between airports based on user inputs. Results are presented using a sophisticated graphical interface. The moderate-sized project is 13,884 lines long in total.

The Kelpie flight planner software makes calculations involving distances, velocities, speeds, accelerations, angles, time and so on, and it does so using a variety of units. Clearly, the software is of the type for which dimensional and units analysis has the potential to discover faults.

 The Kelpie flight planner software is 13,884 lines long, is organized as 10 packages, and is contained in 126 source files. The total number of identifiers in the software is 28,754.

In this case study, the real-world type system for the project contains 35 real-world types, 97 real-world type rules, and 255 real-world typing bindings.

Variables in 32 source files have been interpreted with real-world types. Real-world types have been accessed by program elements inside 50 source files. The other source files do not interact with real-world entities. They do not have real-world type bindings.

## OpenMap Project

OpenMap [58]. OpenMap is a JavaBean-based toolkit for building applications and applets needing geographic information. Using OpenMap components, users can access data from legacy applications. The core components of OpenMap are a set of Swing components that understand geographic coordinates. These components allow users to show map data and manipulate that data. The large-sized project has 157,858 lines of code.

Similar to the Kelpie flight planner, the OpenMap software makes calculations involving distances, heights, speeds, angles, time and so on, and it does so using a variety of units. Clearly, the software is of the type for which real-world constraint checking has the potential to discover units related errors.

# Empirical Study

(Results from dissertation page 144 and onward.)

## Study Process

Overview of what was done and how.

## Result Summary

Tables indicating what was found.

## Example Faults Located

Usual example style.

# Conclusion

##### References

1. L. Jiang, and Z. Su. “Osprey: a practical type system for validating dimensional unit correctness of C programs,” in Proceedings of the 28th international conference on Software engineering (ICSE '06). ACM, New York, NY, USA, 2006, pp. 262-271.
2. Antoniu, T., P. A. Steckler, S. Krishnamurthi, E. Neuwirth, and M. Felleisen. 2004. “Validating the Unit Correctness of Spreadsheet Programs”. In Proceedings of the 26th International Conference on Software Engineering (ICSE), Edinburgh, Scotland, 2004, 439-448. IEEE Computer Society, 2004.
3. Delft, V. 1999. “A Java extension with support for dimensions”. In Softw. Pract. Exper., 605-616. John Wiley & Sons, Inc., 1999
4. Mars Climate Orbiter Mishap Investigation Board Phase I Report, 1999. National Aeronautics and Space Administration, Washington DC, November 10, 1999.