FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Visualization of program execution in gforth

Proposal

BACHELOR'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science

in

Software & Information Engineering

by

Mario Gastegger
Registration Number 0726289

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Martin Ertl

Abstract

Problem definition

In software engineering, an important part is to verify the functional correctness of a program. The difficulty of this task grows with the size and the complexity of a program. Thus the task of finding faulty code consumes a considerable amount of time and the efficiency of finding, understanding and fixing this faulty code is of major concern. A very efficient way to keep up code quality is to make developers understand how the code really works. In this thesis I'm going to implement a visualization of the program execution of Forth programs in Gforth and will analyse the improvement of user experience during debugging and time consumption of the process with example programs.

Expected results

Improvement of awareness of what's happening during the execution of a program and efficiency of finding faulty code.

Methodology and approach

As a first step I'm going to evaluate means of implementing a transparent way to generate a program trace by modification of the Gforth code. The next step is to visualize manipulation of the stack and accessed memory. Once a satisfying visualization in implemented, I'm going to write and debug example programs with an without the visualization to verify my assumption.

State of the art

Current methods of locating faults are

Print debugging

Words like `.`, `.`, `."` and `~~` print information directly to the terminal.

Gforth debugger

Stepping throw program execution with dbg.

Writing test cases

Writing test cases for words to narrow down the actual location of the fault.

Relation to Software engineering

- Software quality assurance (testing, dynamic analysis, debugging)
- Software development methodology (prototyping, agile)
- Stack-based language(forth)

Timetable

Calendar week	work
2014 - 40	Research on forth, the architecture of gforth, “debugging” in forth/gforth, program execution/trace visualization and on similar approaches
2014 - 44	Extracting several technical approaches to accomplish the task(hooks, word-wrapping, level of implementation, ...)
2014 - 45	Evaluation of the approaches(automation, performance, feasibility)
2014 - 46	Prototyping the approaches in order of quality
2014 - 50	Evaluation visualization methods
2014 - 52	Verification of the hypothesis
2015 - 03	Final documentation and feedback cycles
2015 - 05	Submission

References TODO

- print debugging
- test driven development
- visualization of program execution/traces
- debugger