KTH Computer Science
and Communication

Test-oriented runtime verification

Using a test-like specification syntax for runtime verification

ADAM RENBERG

Master's Thesis at CSC
Supervisor Valtech: Erland Ranvinge
Supervisor CSC: Narges Khakpour
Examiner: Tjohej

TRITA xxx yyyy-nn

Abstract

This is a skeleton for KTH theses. More documentation regarding the KTH thesis class file can be found in the package documentation.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Mauris purus. Fusce tempor. Nulla facilisi. Sed at turpis. Phasellus eu ipsum. Nam porttitor laoreet nulla. Phasellus massa massa, auctor rutrum, vehicula ut, porttitor a, massa. Pellentesque fringilla. Duis nibh risus, venenatis ac, tempor sed, vestibulum at, tellus. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos.

Keywords:

Referat

"Test-orienterad runtime-verifiering"

Denna fil ger ett avhandlingsskelett. Mer information om LaTeX-mallen finns i dokumentationen till paketet.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Mauris purus. Fusce tempor. Nulla facilisi. Sed at turpis. Phasellus eu ipsum. Nam porttitor laoreet nulla. Phasellus massa massa, auctor rutrum, vehicula ut, porttitor a, massa. Pellentesque fringilla. Duis nibh risus, venenatis ac, tempor sed, vestibulum at, tellus. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos.

Keywords (Sökord? Nyckelord?):

Preface

This is a master thesis / exjobb in Computer Science at the Royal Institute of Technology (KTH), Stocholm. The work was done at Valtech Sweden, an IT Consultancy. It was supervised by Erland Ranvinge (Valtech) and Dr. (TODO: check) Narges Khakpour (CSC KTH).

Thanks to people.

Contents

Chapter 1

Introduction

This is the introduction.

Purpose: Test-like syntax of runtime verification specifications.

What will this report discuss? What problems? Why is this interesting?

What will this report **not** discuss?

Perhaps: Discuss the sectioning of this report.

Chapter 2

Background

Previous work, in correctness and RV.

2.1 Proving Correctness

2.1.1 Formal Verification

Best result. Tedious. Often impossible.

2.1.2 Model Checking

Nice, simpler than formal verification. Can yield impossibly large state spaces.

2.1.3 Testing

Not formal - doesn't prove anything except for the specified test cases.
Manual. Automatic test-generation?

2.2 Runtime Verification

The idea: Lightweight formal verification. Execution trace. Speed? Monitoring.

2.2.1 Writing Specifications

LTL. TLTL. EAGLE?

2.2.2 Transforming Specifications and Instrumenting Code

Büchi Automatons. AspectJ.

2.2.3 Online v. Offline

Chapter 3

Test-Oriented Runtime Verification

What have I done, and why (again)?

3.1 Testing Frameworks

How do they work? What are their syntaxes?

3.2 Comparing Testing Frameworks, Languages and Environments

Why this testing framework as starting point? Why this language?

3.3 pyrv

3.3.1 General

3.3.2 Syntax?

3.3.3 Correctness

3.4 Conclusions

It is all awwwesomee!

Chapter 4

Discussion

What do we see in the future? How can this be extended, continued?
Results (un)expected? Larger context.
Some speculation? Recommendations?

Appendix A

RDF

And here is a figure

Figure A.1. Several statements describing the same resource.

that we refer to here: A.1