User's Guide for the Design and Testing System YEROTH_QVGE (YR_QVGE)

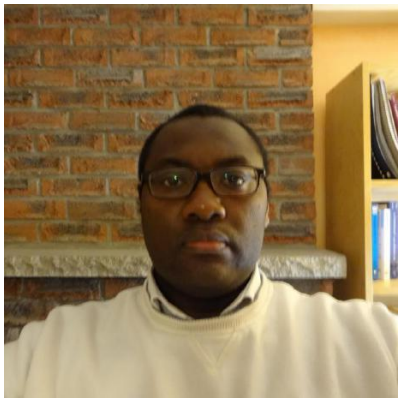


Figure 1: Portrait of PROF. DR.–ING. DIPL.–INF. XAVIER NOUMBISSI NOUNDOU .
Contact: yeroth.d@gmail.com

Table 1: STATE DIAGRAM MEALY MACHINE SPECIFICATION KEYWORDS in YEROTH_QVGE

scientific keywords	engineering keywords
STATE	STATE
START_STATE	BEGIN_STATE
FINAL_STATE	ERROR_STATE
IN_PRE	IN_BEFORE
IN_POST	IN_AFTER
IN_SET_TRACE	IN_SQL_EVENT_LOG
NOT_IN_PRE	NOT_IN_BEFORE
NOT_IN_POST	NOT_IN_AFTER
NOT_IN_SET_TRACE	NOT_IN_SQL_EVENT_LOG

Figure 2: **A SAMPLE state diagram mealy machine file.**

```
1. yr_sd_mealy_automaton_spec yr_missing_department_NO_DELETE
2. {
3.  START_STATE(d):NOT_IN_BEFORE(YR_ASSET,department.department_name)
4.    ->[in_sql_event_log('DELETE.departement.YR_ASSET',STATE(d))]/'SELECT.department'->
5.     ERROR_STATE(e):IN_AFTER(YR_ASSET,stocks.department_name).
6. }
```

Figure 3: A SCREENSHOT OF YEROTH_QVGE.

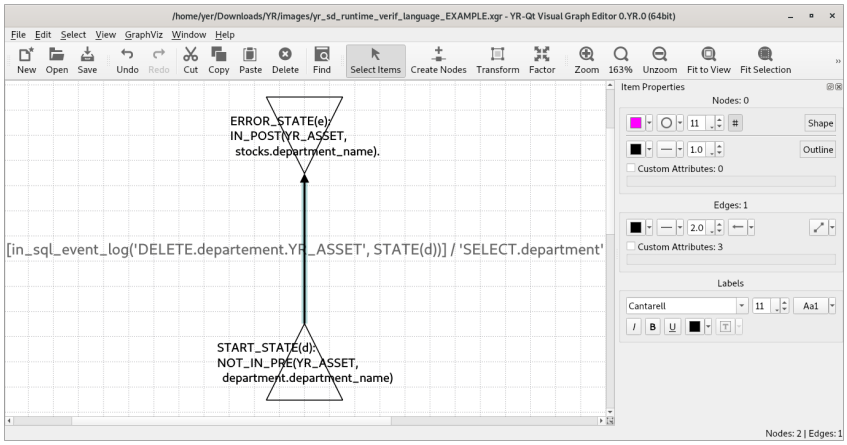
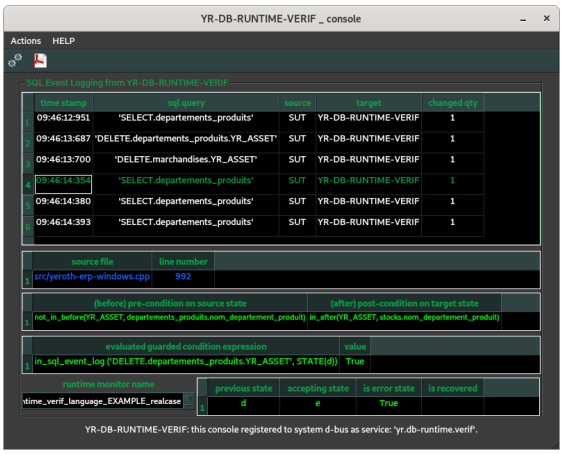


Figure 4: A SCREENSHOT OF YR-DB-RUNTIME-VERIF SQL EVENT LOG.

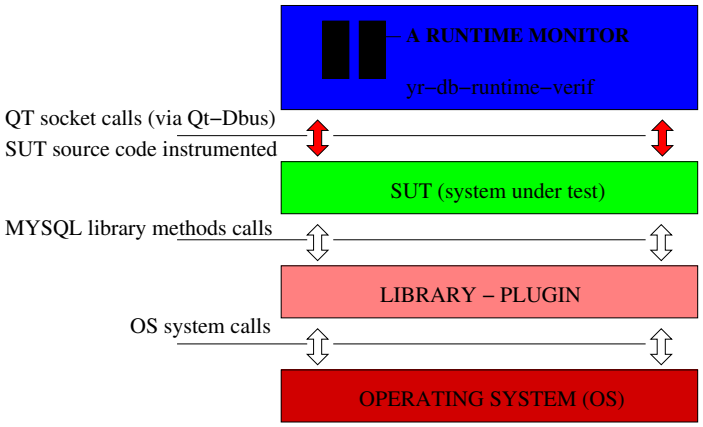


1 Introduction

This user's guide helps briefly and concisely how to create a binary executable of the runtime monitoring testing tool **YR-DB-RUNTIME-VERIF** having user defined runtime monitors. The guide also specifies keywords allowed within runtime monitor specifications as State Diagram Mealy Machines.

2 YEROTH_QVGE (YR_QVGE) Short Overview

Figure 5: SOFTWARE ARCHITECTURE OF YR-DB-RUNTIME-VERIF.



YEROTH_QVGE is a CASE (Computer-Aided Software Engineering) design tool to generate "domain-specific language (DSL) YR_SB_RUNTIME_VERIF_LANG [1]" files, to be inputted into the "compiler YR_SB_RUNTIME_VERIF_LANG_COMP ", so to generate C++ files for the "runtime verifier tester YR-DB-RUNTIME-VERIF [2]" that allows for manual verification of SQL correctness properties of Graphical User Interface (GUI) software.

YR-DB-RUNTIME-VERIF inputs SQL correctness properties expressed using the formalism "*state diagram mealy machine* (YR_SD_RUNTIME_VERIF)". Figure 5 illustrates a software system architecture of YR-DB-RUNTIME-VERIF, together with the monitored program under analysis. The Free Open Source Code Software (FOSS) tool-chain of development testing is located as follows for free, EXCEPT for "YEROTH_QVGE " that is a Closed Source Code Software (CSCS):

- COMPILER YR_SB_RUNTIME_VERIF_LANG_COMP (i.e.: YR_SB_RUNTIME_VERIF_LANG): https://github.com/yerothd/yr_sd_runtime_verif_lang
- RUNTIME VERIFIER TESTER YR-DB-RUNTIME-VERIF: https://github.com/yerothd/yr-db-runtime-verif
- state diagram mealy machine UNIT TESTS CODE (i.e.: YR_SD_RUNTIME_VERIF): https://github.com/yerothd/yr_sd_runtime_verif_UNIT_TESTS
- state diagram mealy machine (i.e.: YR_SD_RUNTIME_VERIF): https://github.com/yerothd/yr_sd_runtime_verif

3 State Diagram Mealy Machine Specification Keywords

Figure 6: A motivating example, as current bug in YEROTH–ERP–3.0.

$\overline{Q0}$:= NOT_IN_BEFORE(YR_ASSET, department.department_name).
$\overline{Q1}$:= IN_AFTER(YR_ASSET, stocks.department_name).

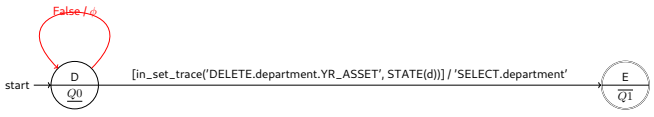


TABLE 1 depicts scientific keywords and their engineering counterpart that can be used in describing NOT DESIRABLE [3] SQL [4] call sequence state diagram mealy machine in YEROTH_QVGE Design and Testing System.

[1] https://github.com/yerothd/yr_sd_runtime_verif_lang
[2] https://github.com/yerothd/yr-db-runtime-verif
[3] Scientific: fail (forbidden) trace.
[4] Structure Query Language.

A STATE DIAGRAM mealy machine specification is compiled into C++ code that describes a runtime monitor to be executed in the runtime monitoring tester YR-DB-RUNTIME-VERIF. Figure 2 depicts a sample State Diagram Mealy Machine specification on a NOT DESIRABLE SQL call sequence. Figure 6 shows a finite automaton representation of the mealy machine description in Figure 2.

4 Custom User Project

Table 2: YR-DB-RUNTIME-VERIF Directories

Variable for illustration purposes	Meaning
$YR−DB−RUNTIME−VERIF	root directory of YR-DB-RUNTIME-VERIF
$YR−DB−RUNTIME−VERIF/$USER_PROJECT_DIR	root directory of user project

Table 2 illustrates directories that will be used to describe a process to generate a single binary executable for a user's custom project with several runtime monitor specifications.

Creating a binary executable for State Diagram Mealy Machine (SDMM) specifications consists of the following elements:

- Property configuration file: this file defines environment variables necessary for building a binary executable for the user; it is located in path: **"$USER_PROJECT_DIR/bin/configuration-properties.sh"**.
- **"$USER_PROJECT_DIR/sd-mealy-machine-specs"**: this directory contains user defined State Diagram Mealy Machine (SDMM) specifications to generate Corresponding runtime monitors within a single binary executable.
- Generate an executable for a user defined runtime monitor:

  a) execute following command in directory **"$USER_PROJECT_DIR"**:

  ```
  ./YR-create-executable-for-user-SDMM.sh -d $USER_PROJECT_DIR
  ```

  b) modify the LTL verification code part within the generated source code files.

  Then execute following command in directory **"$USER_PROJECT_DIR"**:

  ```
  ./yr_db_runtime_verif_BUILD_DEBIAN_PACKAGE.sh
  ```

  c) uninstall YR-DB-RUNTIME-VERIF with following command in directory **"$USER_PROJECT_DIR"**:

  ```
  ./yr_DB_RUNTIME_VERIF_uninstall.sh
  ```

  d) re−install YR-DB-RUNTIME-VERIF with following command in directory **"$USER_PROJECT_DIR"**:

  ```
  ./yr_DB_RUNTIME_VERIF_INSTALL.SH
  ```

  The generated binary executable ('yr-db-runtime-verif') appears in directory **"$YR−DB−RUNTIME−VERIF/bin"**.

5 Formal Scientific and Engineering Project Description

Detailed formal scientific and engineering contributions of design and testing system YEROTH_QVGE can be found in ***JOURNAL ARTICLE "Runtime Verification Of SQL Correctness Properties with YR-DB-RUNTIME-VERIF"*** at: https://zenodo.org/record/8381187.

6 Conclusion

YEROTH_QVGE costs only 3,000 EUROS. WE ONLY SUPPORT DEBIAN−LINUX (https://www.debian.org).

Information Brochure of the Design and Testing System YEROTH_QVGE (YR_QVGE)

PROF. DR.–ING. DIPL.–INF. XAVIER NOUMBISSI NOUNDOU
CONTACT: yeroth.d@gmail.com

Table 1: EQUIVALENCES

scientific literature	engineering acronym
PRE	BEFORE
POST	AFTER
A TRACE	AN EVENT LOG
A FINAL STATE	AN ERROR STATE

Figure 1: **A SAMPLE state diagram mealy machine file.**

```
1. yr_sd_mealy_automaton_spec yr_missing_department_NO_DELETE
2. {
3.  START_STATE(d):NOT_IN_BEFORE(YR_ASSET,department.department_name)
4.   ->[in_sql_event_log('DELETE.departement.YR_ASSET',STATE(d))]/'SELECT.department'->
5.    ERROR_STATE(e):IN_AFTER(YR_ASSET,stocks.department_name).
6. }
```

Figure 2: A SCREENSHOT OF YEROTH_QVGE.

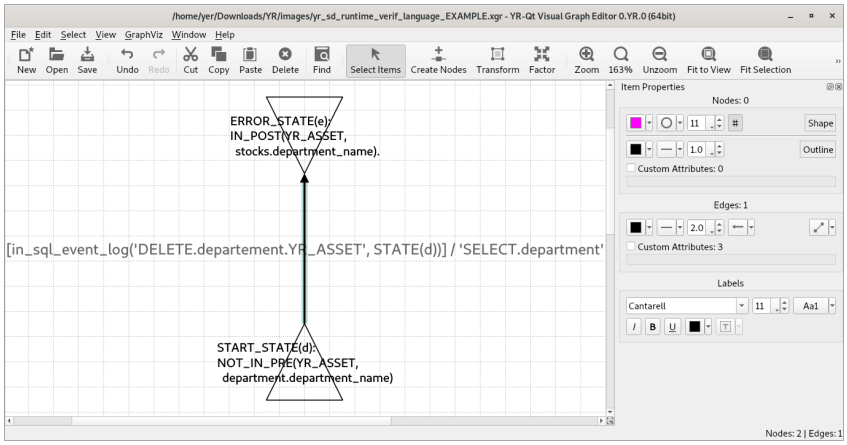
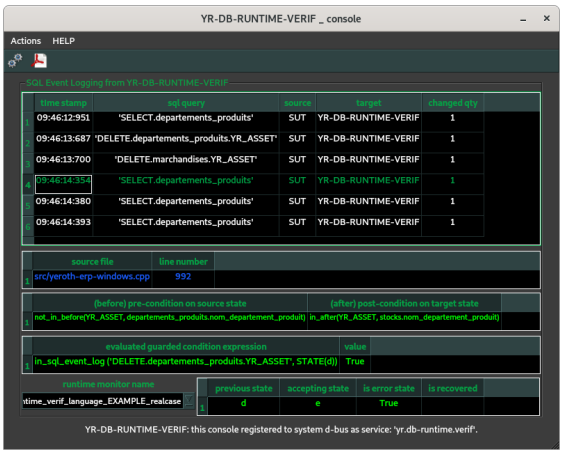


Figure 3: A SCREENSHOT OF YR-DB-RUNTIME-VERIF SQL EVENT LOG.



1 Developer Biography

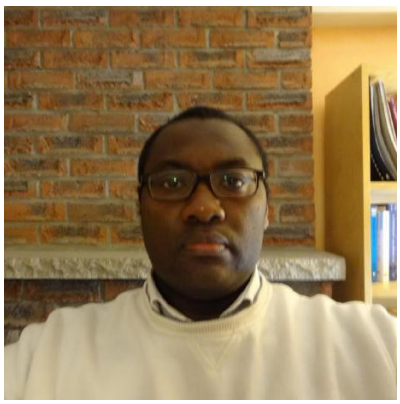


Figure 4: Portrait of XAVIER.

PROF. DR.–ING. DIPL.–INF. XAVIER NOUMBISSI NOUNDOU is a CHRISTIAN BY FAITH, Cameroonian, born on September 16 1983 in DOUALA (LITTORAL region, CAMEROON). Xavier has a *"Diplom–Informatiker (Dipl.–Inf.)"* qualification from the **University of Bremen, Bremen, Bremen, GERMANY** (May 25, 2007). XAVIER NOUMBISSI NOUNDOU IS A PHILOSOPHIAE DOCTOR (PH.D.) from **THE UNIVERSITY OF WATERLOO (ON, CANADA); DECEMBER** 20**,** 2011**!**

PROF. DR.–ING. DIPL.–INF. XAVIER NOUMBISSI NOUNDOU has worked together with **PROF. DR. RER. NAT. HABIL. jan peleska**, at AGBS–University of Bremen, GERMANY; and 2 years later at WatForm (Waterloo Formal Methods, ON, Canada) with **PATRICK LAM, P.Eng (Ontario, CANADA), PH.D. (MIT, BOSTON, MA, USA)**.

Xavier has following academic and professional engineering research contributions:

1. 'Context-Sensitive Staged Static Taint Analysis For C using LLVM'
   1. source code in C++: `https://github.com/sazzad114/saint`
   2. full text: `https://zenodo.org/record/8051293`
2. 'YEROTH-ERP-3.0':
   1. source code in C++:
      a. YEROTH–ERP–3.0: `https://github.com/yerothd/yeroth-erp-3-0`
      b. YEROTH–ERP–3.0 SYSTEM DAEMON: `https://github.com/yerothd/yeroth-erp-3-0-system-daemon`
   2. full text (ongoing publication): `https://zenodo.org/record/8052724` .
3. 'Statistical test case generation for reactive systems' at RTT-MBT at (https://www.verified.de) Verified Systems International GmbH.

2 Introduction

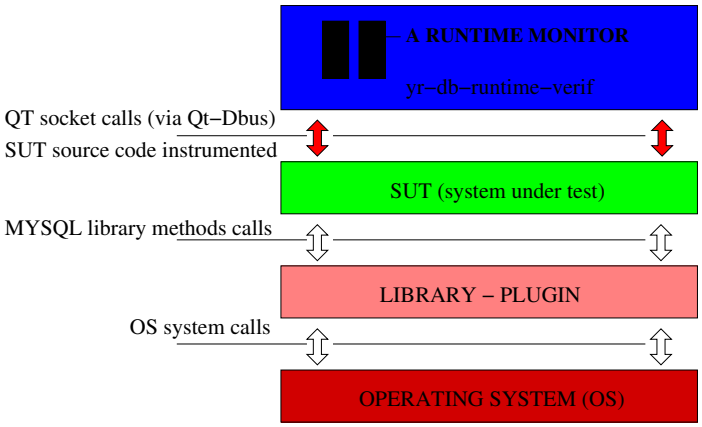


Figure 5: SOFTWARE ARCHITECTURE OF YR-DB-RUNTIME-VERIF.

YEROTH_QVGE is a CASE (Computer-Aided Software Engineering) design tool to generate "domain-specific language (DSL) YR_SB_RUNTIME_VERIF_LANG [1]" files, to be inputted into the "compiler YR_SB_RUNTIME_VERIF_LANG_COMP ", so to generate C++ files for the "runtime verifier tester YR-DB-RUNTIME-VERIF [2]" that allows for manual verification of SQL correctness properties of Graphical User Interface (GUI) software.

YR-DB-RUNTIME-VERIF inputs SQL correctness properties expressed using the formalism "*state diagram mealy machine* (YR_SD_RUNTIME_VERIF)". Figure 5 illustrates a software system architecture of YR-DB-RUNTIME-VERIF, together with the monitored program under analysis. The Free Open Source Code Software (FOSS) tool-chain of development testing is located as follows for free, EXCEPT for "YEROTH_QVGE " that is a Closed Source Code Software (CSCS):

- COMPILER YR_SB_RUNTIME_VERIF_LANG_COMP (i.e.: YR_SB_RUNTIME_VERIF_LANG): `https://github.com/yerothd/yr_sd_runtime_verif_lang`
- RUNTIME VERIFIER TESTER YR-DB-RUNTIME-VERIF: `https://github.com/yerothd/yr-db-runtime-verif`
- state diagram mealy machine UNIT TESTS CODE (i.e.: YR_SD_RUNTIME_VERIF): `https://github.com/yerothd/yr_sd_runtime_verif_UNIT_TESTS`
- state diagram mealy machine (i.e.: YR_SD_RUNTIME_VERIF): `https://github.com/yerothd/yr_sd_runtime_verif`

3 Advantages of YEROTH_QVGE

A sample state diagram mealy machine is shown in Figure 1.

WITH manual drawing of SQL CORRECTNESS PROPERTY MODEL, you are freed from manually writing "state diagram mealy machine text files" that could be tedious and lengthy. Also, editing state diagram mealy machine files manually could be more error-prone than letting a compiler (YR_SB_RUNTIME_VERIF_LANG_COMP) do it for you.

4 Conclusion

YEROTH_QVGE costs only 3,000 EUROS. WE ONLY SUPPORT DEBIAN–LINUX (`https://www.debian.org`).

[1] `https://github.com/yerothd/yr_sd_runtime_verif_lang`
[2] `https://github.com/yerothd/yr-db-runtime-verif`