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	END_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 motivating example, as previous bug found in YEROTH–ERP–3.0.

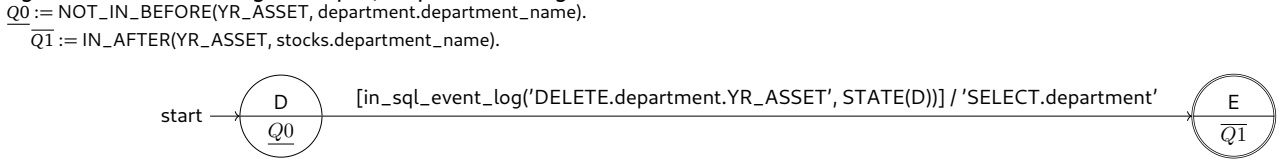


Figure 3: **A SAMPLE state diagram mealy machine file.**

```
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 4: A SCREENSHOT OF YEROTH_QVGE.

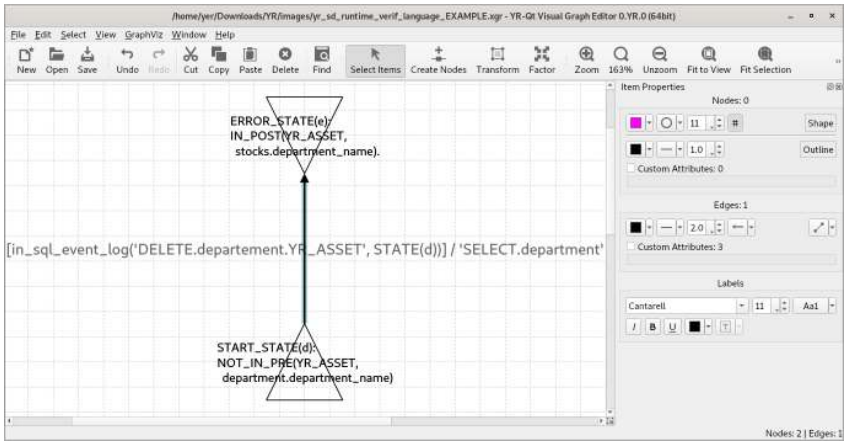
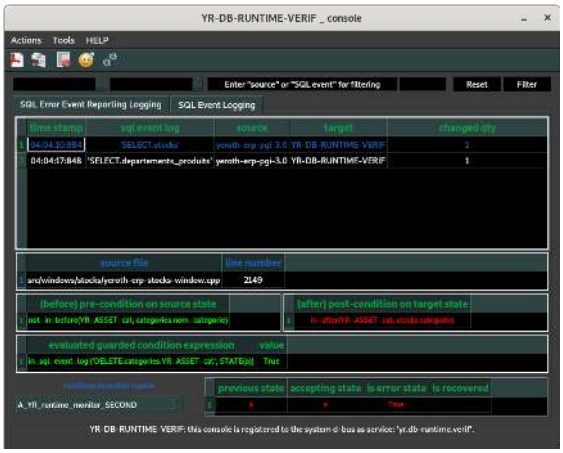
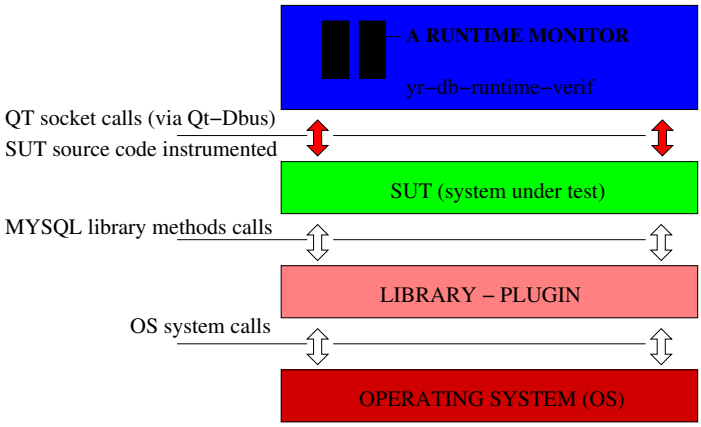


Figure 5: A SCREENSHOT OF YR-DB-RUNTIME-VERIF SQL EVENT LOG.



1 Introduction

Figure 6: SOFTWARE ARCHITECTURE OF YR-DB-RUNTIME-VERIF.



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.

YEROTH_QVGE (YR_QVGE) could be used for the following automatic generation, analysis, verification, and validation tasks:

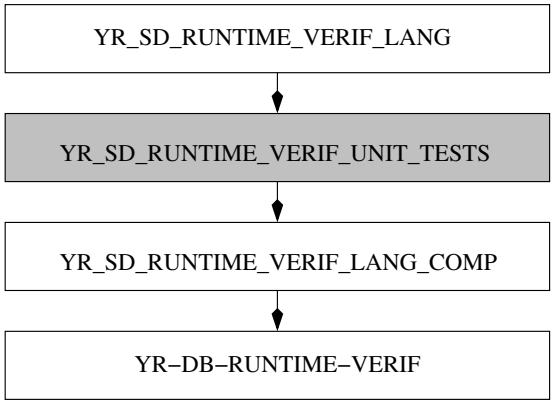
1. Automatic generation of runtime monitoring module program for any software that can emit DBus messages.

   Such runtime monitoring modules are for interest for special LTL model checking properties that cannot get a definite answer through use of a conventional model checker.

2. Software design properties with SQL
3. Software design properties including event sequences over different layers of software system architecture
4. Class diagram with sequence diagram.

2 YEROTH_QVGE (YR_QVGE) Short Overview

Figure 7: YEROTH_QVGE software library dependencies.



YEROTH_QVGE is a CASE (Computer-Aided Software Engineering) design tool to generate "domain-specific language (DSL) YR_SD_RUNTIME_VERIF_LANG [1]" files, to be inputted into the "compiler YR_SD_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.

Figure 8 illustrates a workflow diagrammatically of the afore described process.

Figure 7 show a diagram of the afore described process; The step of the unit tests is colored in gray because it is only for developers of YEROTH_QVGE intended.

YR-DB-RUNTIME-VERIF inputs SQL correctness properties expressed using the formalism "*state diagram mealy machine* (YR_SD_RUNTIME_VERIF_LANG)". Figure 6 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

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

for "YEROTH_QVGE " that is a Closed Source Code Software (CSCS):

- COMPILER (i.e.: YR_SD_RUNTIME_VERIF_LANG_COMP): https://github.com/yerothd/yr_sd_runtime_verif_lang
- RUNTIME VERIFIER TESTER (i.e.: 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_UNIT_TESTS): https://github.com/yerothd/yr_sd_runtime_verif_UNIT_TESTS
- state diagram mealy machine (i.e.: YR_SD_RUNTIME_VERIF_LANG): https://github.com/yerothd/yr_sd_runtime_verif

3 YEROTH_QVGE (YR_QVGE) Project Dependency

Table 2: YEROTH_QVGE Design and Testing System Dependencies

PROJECT	Required Library
1) YR_SD_RUNTIME_VERIF_LANG	
2) YR_SD_RUNTIME_VERIF_LANG_COMP	1)
3) YR_SD_RUNTIME_VERIF_UNIT_TESTS	1)
4) YR-DB-RUNTIME-VERIF	2)

Table 2 illustrates for each library project, which others it depends on.

4 Advantages of YEROTH_QVGE

A sample state diagram mealy machine is shown in Figure 3.

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_SD_RUNTIME_VERIF_LANG) do it for you.

5 State Diagram Mealy Machine (SDMM)

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.

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 3 depicts a sample State Diagram Mealy Machine specification on a NOT DESIRABLE SQL call sequence.

[3] Scientific: fail (forbidden) trace.
[4] Structure Query Language.

5.1 HOW TO READ A "SDMM"

Figure 2 shows a finite automaton representation of the mealy machine description in Figure 3. It shall be read as follows:

- The program is in a start state D; state D is a start state since there is incoming "START" arrow into it.
- (**Pre-**) Condition $\overline{Q0}$: "department name 'YR_ASSET' is not in table column 'department_name' of database table 'department' "; applies in state D.
- Whenever GUARD CONDITION : ***in_sql_event_log('DELETE.department.YR_ASSET', STATE(d))***: "event 'DELETE.department.YR_ASSET' appears in SQL event log (trace) leading to state D"; applies in state D, system under test (SUT) event 'SELECT.department' could occur.
- When SUT event 'SELECT.department' occurs, SUT is now in state E; state E is an error state because the node that represents it in Figure 2 has 2 circles on it.
- (**Post**-) Condition $\overline{Q1}$: "department name 'YR_ASSET' is in table column 'department_name' of database table 'stocks' "; applies in state E.

  This shall not be the case since department 'YR_ASSET' is no more defined in SUT database table 'department'.

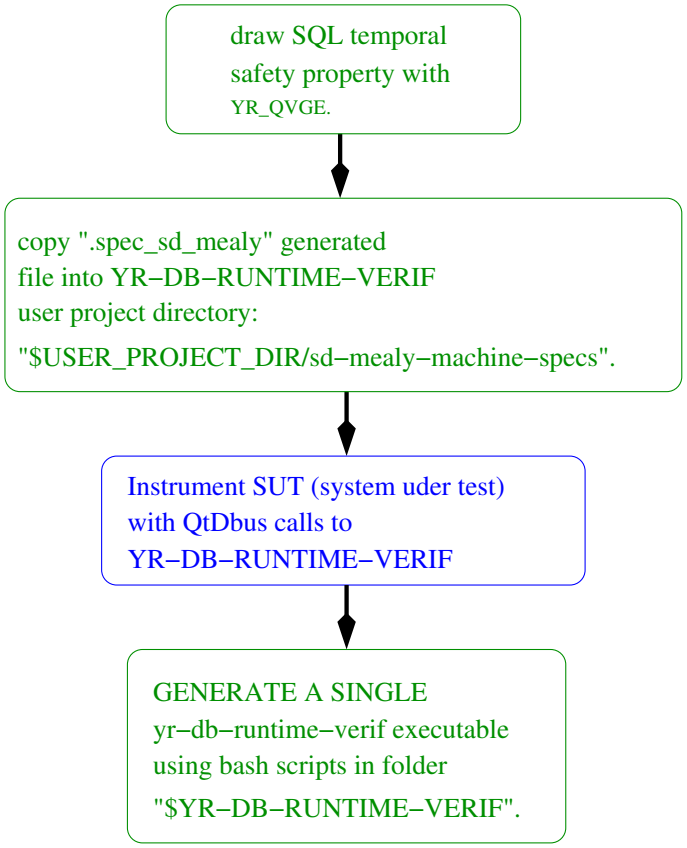
5.2 "SDMM" WITH MORE THAN 2 STATES

State Diagram Mealy Machines (SDMM) with more than 2 states have following characteristics, as detailed in scientific and engineering journal paper [Nou23] in preparation:

- Only the first transition has a pre-condition specification
- Each other transition only has a post-condition specification
- Since each state only has 1 outgoing state transition, the post-condition of the previous (incoming) state transition acts as the pre-condition of the next transition.

6 YEROTH_QVGE (YR_QVGE) Workflow

Figure 8: Workflow.



The **"Design and Testing System"** YEROTH_QVGE works with following workflow, as illustrated graphically in Figure 8:

1. Draw Structure Query Language (SQL) temporal safety property using drawing tool YEROTH_QVGE;
2. copy the generated ".spec_sd_mealy" files into a user project directory in YR-DB-RUNTIME-VERIF home development folder: "$YR–DB–RUNTIME–VERIF";
3. follow the steps described in Section 7 so to gather a single executable that defines all specified runtime monitors.

7 Custom User Project (YR–DB–RUNTIME–VERIF)

Table 3: 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	root directory of user project

Table 3 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.

Figure 5 illustrates a screenshot of the Graphical User Interface (GUI) of YR-DB-RUNTIME-VERIF . You can get a copy of YR-DB-RUNTIME-VERIF using the following command:

git clone https://github.com/yerothd/yr-db-runtime-verif

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

1. 'MariaDB' database connection configuration file: this file defines settings to connect to the system under test (SUT) application database; it is located in path: **"$YR–DB–RUNTIME–VERIF/YR-DB-RUNTIME-VERIF-GUI-ELEMENTS-SETUP/yr-db-runtime-verif-database-connection.properties"**.

   A database connection to the SUT application database is required in order to check LTL property through the SDMM application library **YR_SD_RUNTIME_VERIF_LANG .**

2. Property configuration file: this file defines environment variables necessary for building a binary executable for the user; it is located in path: **"$YR–DB–RUNTIME–VERIF/$USER_PROJECT/bin/configuration-properties.sh"**.

3. **"$YR–DB–RUNTIME–VERIF/$USER_PROJECT/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.

4. Generate an executable for a user defined runtime monitor:

   a) execute following command in directory **"$YR–DB–RUNTIME–VERIF"**:

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

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

   Then execute following command in directory **"$YR–DB–RUNTIME–VERIF"**:

   ```
   ./yr_db_runtime_verif_BUILD_DEBIAN_PACKAGE.sh
   ```

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

   ```
   ./yr_DB_RUNTIME_VERIF_uninstall.sh
   ```

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

   ```
   ./yr_DB_RUNTIME_VERIF_INSTALL.SH
   ```

8 HOW TO START YR-DB-RUNTIME-VERIF

- The "ELF-x64" binary executable, in the source development directory is located in full path: "$YR-DB-RUNTIME-VERIF/bin".

- The DEBIAN–LINUX icon ( ) of YR-DB-RUNTIME-VERIF is located in "Applications" menu under section "Programming", and section "Accessories".

- The "ELF-x64" binary executable, after installation of the DEBIAN–LINUX package 'yr-db-runtime-verif.deb' is located in full path: "/opt/yr-db-runtime-verif/bin".

9 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"*** [Nou23].

10 Conclusion

The graphical drawing tool YEROTH_QVGE (Figure 4) costs only 3,000 EUROS. WE ONLY SUPPORT DEBIAN–LINUX (https://www.debian.org).

References

[Nou23] Xavier N. Noundou. Runtime Verification Of SQL Correctness Properties with YR-DB-RUNTIME-VERIF. https://zenodo.org/record/8381187, October 2023.

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 motivating example, as previous bug found in YEROTH–ERP–3.0.

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

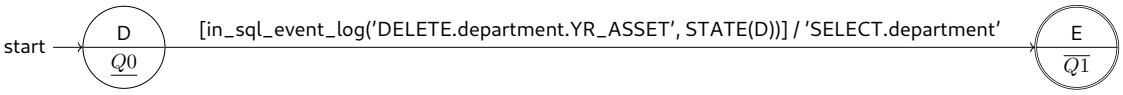


Figure 2: **A SAMPLE state diagram mealy machine file.**

```
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.

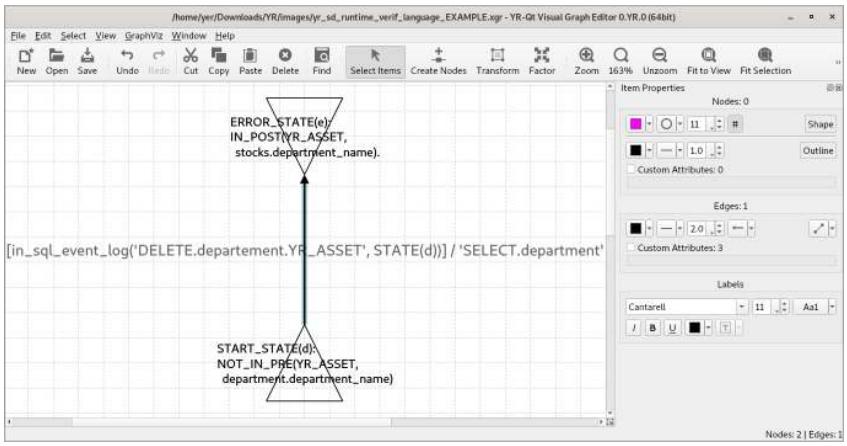
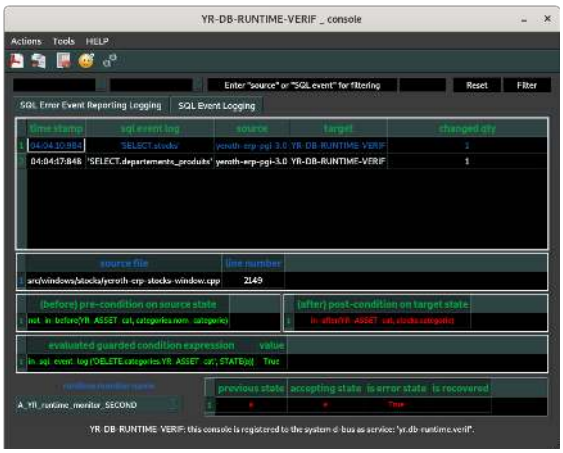


Figure 4: A SCREENSHOT OF YR-DB-RUNTIME-VERIF SQL EVENT LOG.



1 Developer Biography

Figure 5: 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 (and at spin–off VERIFIED SYSTEMS INTERNATIONAL GmbH), GERMANY; and 2 years later at WatForm–University of Waterloo, ON, Canada, with **PATRICK LAM, PH.D. (MIT, BOSTON, MA, USA), P.ENG. (Ontario, CANADA)**.

Xavier could successfully work with **DR. FRANK TIP** at The University of Waterloo (Waterloo, ON, Canada) on his first JAVA dynamic program analysis.

Xavier also had the great opportunity through **DR. MARCEL MITRAN** and **PATRICK LAM, PH.D., P.ENG.**; to work as a graduate intern in Markham (Toronto, ON, CANADA) at IBM TORONTO SOFTWARE LABORATORY; in the JAVA–J9 Just–In–Time Compiler Optimization Team, together with **VIJAY SUNDARESAN**.

Xavier has following academic and professional engineering research contributions:

1. 'Statistical test case generation for reactive systems' at RTT-MBT at VERIFIED SYSTEMS INTERNATIONAL GmbH (https://www.verified.de).

2. 'Context-Sensitive Staged Static Taint Analysis For C using LLVM'

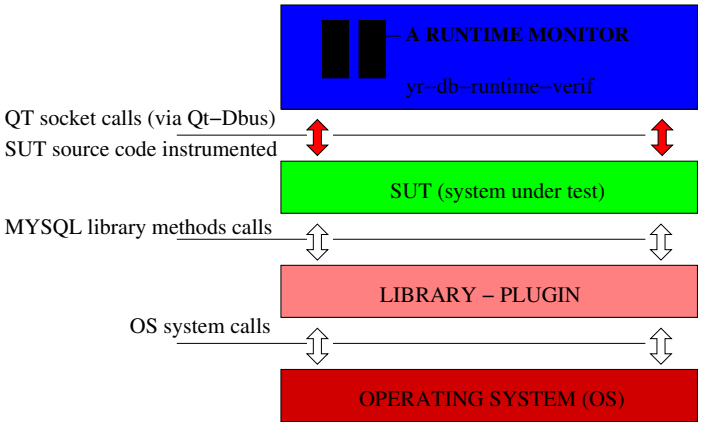
   1. source code in C++:
      `https://github.com/sazzad114/saint`
   2. full text: `https://zenodo.org/record/8051293`

3. '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` .

2 Introduction

Figure 6: 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_SD_RUNTIME_VERIF_LANG [1]" files, to be inputted into the "compiler YR_SD_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_LANG). Figure 6 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 (i.e.: YR_SD_RUNTIME_VERIF_LANG_COMP):
  `https://github.com/yerothd/yr_sd_runtime_verif_lang`
- RUNTIME VERIFIER TESTER (i.e.: 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_UNIT_TESTS):
  `https://github.com/yerothd/yr_sd_runtime_verif_UNIT_TESTS`
- state diagram mealy machine (i.e.: YR_SD_RUNTIME_VERIF_LANG):
  `https://github.com/yerothd/yr_sd_runtime_verif`

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

3 YEROTH_QVGE (YR_QVGE) Project Dependency

Table 2: YEROTH_QVGE Design and Testing System Dependencies

PROJECT	Required Library
1) YR_SD_RUNTIME_VERIF_LANG	
2) YR_SD_RUNTIME_VERIF_LANG_COMP	1)
3) YR_SD_RUNTIME_VERIF_UNIT_TESTS	1)
4) YR-DB-RUNTIME-VERIF	2)

Table 2 illustrates for each library project, which others it depends on.

Figure 7: YEROTH_QVGE software library dependencies.

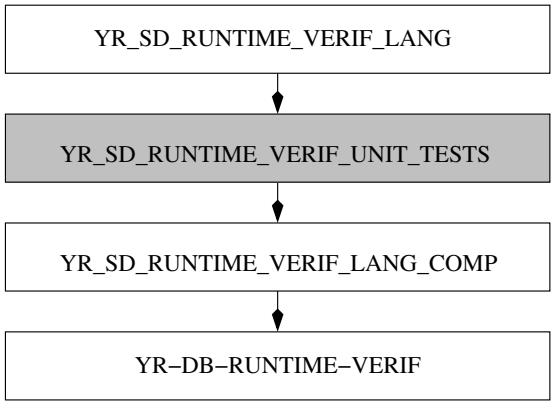


Figure 7 show a diagram overview of the presentation in Table 2. The step of the unit tests is colored in gray because it is only for developers of YEROTH_QVGE intended.

4 Potential Uses of YEROTH_QVGE

YEROTH_QVGE (YR_QVGE) could be used for the following automatic generation, analysis, verification, and validation tasks:

1. Automatic generation of runtime monitoring module program for any software that can emit DBus messages.

   Such runtime monitoring modules are for interest for special LTL model checking properties that cannot get a definite answer through use of a conventional model checker.

2. Software design properties with SQL
3. Software design properties including event sequences over different layers of software system architecture
4. Class diagram with sequence diagram.

5 Advantages of YEROTH_QVGE

A sample state diagram mealy machine is shown in Figure 2.

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_SD_RUNTIME_VERIF_LANG_COMP) do it for you.

6 Conclusion

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

Index