



The ORIS Tool: **App**, **Library** and **Toolkit** for Quantitative Evaluation of Non-Markovian Systems

Laura Carnevali ¹ Marco Paolieri ² Enrico Vicario ¹

¹Department of Information Engineering, University of Florence, Italy

²Department of Computer Science, University of Southern California, USA

TOSME 2021: Workshop on Tools for Stochastic Modelling and Evaluation
November 12, 2021

Extended contents about ORIS are reported in:

The ORIS Tool: Quantitative Evaluation of Non-Markovian Systems

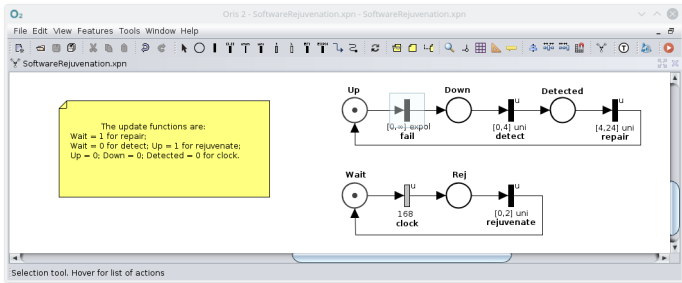
by Marco Paolieri, Marco Biagi, Laura Carnevali, Enrico Vicario

IEEE Trans. on Software Engineering, vol. 47, no. 6, pp. 1211–1225, June 2021

DOI: 10.1109/TSE.2019.2917202

ORIS as a GUI Application for Models of Stochastic Systems

- GUI application for **quantitative evaluation** of **stochastic models**
 - Model specification using Stochastic Time Petri Nets (STPNs)
 - Model validation using interactive simulation (token game)
 - Transient/steady-state model analysis using different analysis engines



Property editor
Edit properties of the selected components

Default properties | Stochastic properties

Type: Exponential

Evaluate Normalization factor: 1.0

EPT:0	LFT:72
Expn: 0.000139	
EPT:72	LFT:1.44
Expn: 0.0000694	
EPT:1.44	LFT:216
Expn: 0.000139	
EPT:216	LFT:
Expn: 0.00347 * Exp[0.00219 s]	

OK Cancel

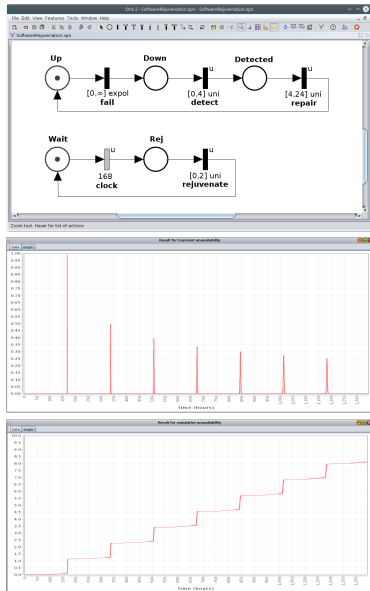
Evaluation of Quantitative Properties in ORIS

• Transient unavailability

- Transient probability that the system is not working at time t
- Instantaneous transient reward $\text{Down} > 0 \mid \text{Detected} > 0 \mid \text{Rej} > 0$ (regenerative transient analysis, time limit 1344 h, step size 0.005 h)

• Cumulative unavailability

- Expected outage time within the time interval $[0, t]$
- Cumulative transient reward $\text{Down} > 0 \mid \text{Detected} > 0 \mid \text{Rej} > 0$ (regenerative transient analysis, time limit 1344 h, step size 0.005 h)



Analysis Engines in ORIS

Different classes of **stochastic processes** supported:

- Only exponential timers: **Continuous-Time Markov Chain** (CTMC)
- General timers reset after each firing: **Semi-Markov Process** (SMP)
- General timers reset after n firings: **Markov Regenerative Process** (MRP)
- No restrictions: **Generalized Semi-Markov Process** (GSMP)

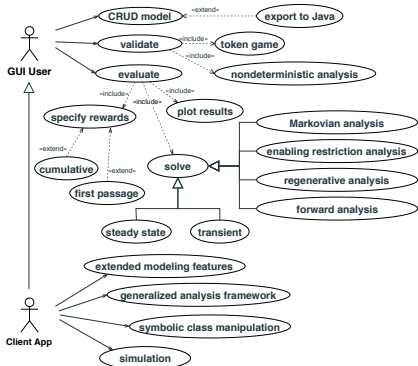
Papers on the **solution techniques** implemented in ORIS:

- *Probabilistic model checking of regenerative concurrent systems*
M. Paolieri, A. Horváth, E. Vicario, IEEE Trans. on Software Engineering, 2016
- *Transient analysis of non-Markovian models using stochastic state classes*
A. Horváth, M. Paolieri, L. Ridi, E. Vicario, Performance Evaluation, 2012
- *Using Stochastic State Classes in Quantitative Evaluation of Dense-Time Reactive Systems*
E. Vicario, L. Sassoli, L. Carnevali, IEEE Trans. on Software Engineering, 2009
- *State-Density Functions over DBM Domains in the Analysis of Non-Markovian Models*
L. Carnevali, L. Grassi, E. Vicario, IEEE Trans. on Software Engineering, 2009
- *Static Analysis and Dynamic Steering of Time-Dependent Systems Using Time Petri Nets*
E. Vicario, IEEE Trans. on Software Engineering, 2001

ORIS as a Java Library

ORIS provides a Java library (SIRIO) for modeling and analysis of STPNs

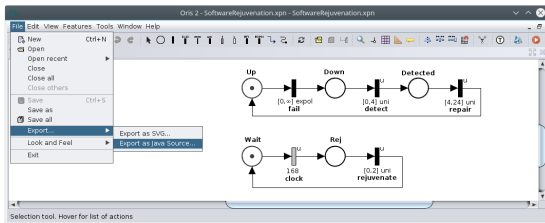
- To carry out **extensive performance and reliability studies**
- To implement **models@runtime**
- To implement new **model features** and new **analysis methods**



```
1 package it.unifi.oris.sirio.valueTools2017Tutorial;
2
3 import java.awt.EventQueue;
4
5 public class TransientAnalysisUnreliabilityLauncher {
6
7     private static final OmegaBigDecimal fRowBound = new OmegaBigDecimal("4000");
8     private static final OmegaBigDecimal fResStep = new OmegaBigDecimal("1e-20");
9
10    private static final String UNRELIABILITY_REWARD = "f((k=0, l=0))";
11    private static final MarkingCondition ABSORPTION_CONDITION = MarkingCondition.fromString("k=0");
12
13    public static void showPlot(TransientSolutionDeterministicEnablingState, RewardRate solution) {
14        final JPanel plot = TransientSolutionViewer.solutionChart(solution);
15        EventQueue.invokeLater(new Runnable() {
16
17            @Override
18            public void run() {
19                JFrame frame = new JFrame("Plot");
20                frame.add(plot);
21                frame.setDefaultCloseOperation(3);
22                frame.setDefaultCloseOperation(0);
23                frame.pack();
24                frame.setLocationRelativeTo(null);
25                frame.setVisible(true);
26            }
27        });
28    }
29
30    public static void main(String[] args) {
31        // Repeat for different values of waitClock.
32        for (int waitClock=100; waitClock<400; waitClock+=100) {
33
34            // 1. Build the model (pn) and the initial marking (m).
35            PetriNet pn = new PetriNet();
36            Marking m = new Marking();
37            SEWjuveneration.build(pn, m, waitClock);
38
39            // 2. Evaluate the unreliability reward.
40            DeterministicEnablingState initialRegeneration = new DeterministicEnablingState(m, pn);
41            RegenerativeTransientAnalysis deterministicEnablingState_unreliabilityAnalysis = RegenerativeTransientAnalysis
42                .compute(pn, initialRegeneration,
43                    new DeterministicEnablingStateBuilder(pn, true),
44                    new EnablingSyncEvaluators(),
45                );
46        }
47    }
48 }
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

ORIS as a Toolkit

- Define/validate model with app and **export it as Java code** that uses SIRIO



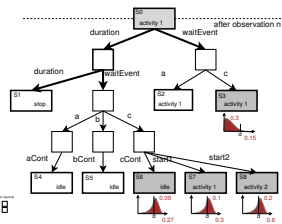
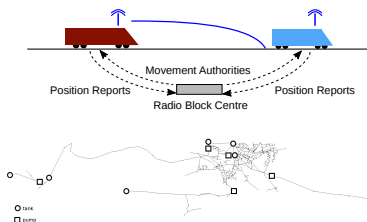
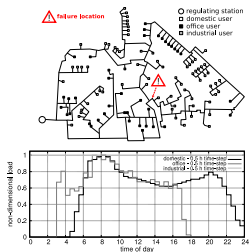
The screenshot shows the IDE interface with the Petri net model exported as Java code. The code is in a file named "SoftwareRegeneration.java". It defines a class "SoftwareRegeneration" with a "main" method. The code uses the SIRIO library to define the Petri net model. The model is defined as a Petri net with five places and five transitions. The places are "Up", "Down", "Detected", "Wait", and "Rej". The transitions are "fail", "detect", "repair", "clock", and "rejuvenate". The code uses the SIRIO library to define the Petri net model and to generate the Java code for the model. The code is as follows:

```
public static void fail(Servlet req, Markup markup) {
    // Place
    Place Up = new Place("Up");
    Place Down = new Place("Down");
    Place Detected = new Place("Detected");
    Place Wait = new Place("Wait");
    Place Rej = new Place("Rej");
    // Transitions
    Transition fail = new Transition("fail");
    Transition detect = new Transition("detect");
    Transition repair = new Transition("repair");
    Transition clock = new Transition("clock");
    Transition rejuvenate = new Transition("rejuvenate");
    // Petri net
    PetriNet pn = new PetriNet("SoftwareRegeneration");
    pn.addPlace(Up);
    pn.addPlace(Down);
    pn.addPlace(Detected);
    pn.addPlace(Wait);
    pn.addPlace(Rej);
    pn.addTransition(fail);
    pn.addTransition(detect);
    pn.addTransition(repair);
    pn.addTransition(clock);
    pn.addTransition(rejuvenate);
    // Initial state
    pn.setInitialState(Up);
    // Simulation
    Simulation sim = new Simulation(pn);
    sim.run();
}
```

Applications using ORIS

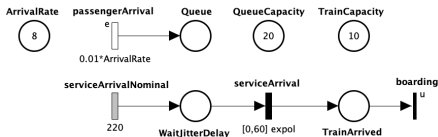
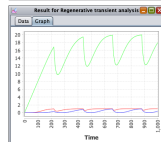
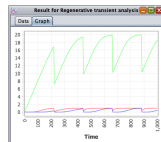
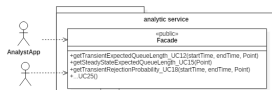
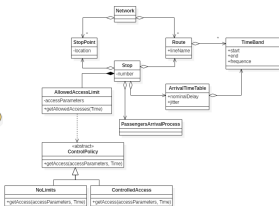
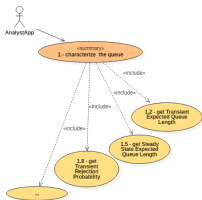
Papers on **applications** of stochastic models analyzed with ORIS:

- *Performability evaluation of the ERTMS/ETCS - Level 3*
M. Biagi, L. Carnevali, M. Paolieri, E. Vicario, Transportation Research: Part C, 2017
- *A Quantitative Approach to Input Generation in Real-Time Testing of Stochastic Systems*
L. Carnevali, L. Ridi, E. Vicario, IEEE Trans. on Software Engineering, 2013
- *Performability Evaluation of Water Distribution Systems During Maintenance Procedures*
L. Carnevali, F. Tarani, E. Vicario, IEEE Trans. on Sys., Man, and Cybern. Systems, 2020
- *Model-based quantitative evaluation of repair procedures in gas distribution networks*
M. Biagi, L. Carnevali, F. Tarani, E. Vicario, ACM Trans. on Cyber-Physical Systems, 2018
- *A continuous-time model-based approach for activity recognition in pervasive environments*
M. Biagi, L. Carnevali, M. Paolieri, F. Patara, E. Vicario, IEEE Trans. Human-Machine Sys., 2019



Using ORIS to support Model Driven Engineering: from Domain Metamodels to Runtime Analysis


- Manual **design of STPN model** and **export as Java** with ORIS app
- Object-oriented **domain model** visited by **model builder**
- Facade to **evaluate metrics** for use cases of **system analysts** and **users**



Obtaining ORIS

- Freely available (source code released under AGPL license)
- Downloads, tutorials, API documentation at www.oris-tool.org
- Java library examples at github.com/oris-tool/sirio-examples

Oris Tool | Home | Tutorial | Sirio Library | Publications | Contributors



UNIVERSITÀ
DEGLI STUDI
FIRENZE

Software Science and Technology Lab

ORIS Tool

Analysis of timed and stochastic Petri nets

[Download](#)

Release 2.3.0 (ask for [support](#)).

Unzip and launch `run.sh` (Linux/macOS) or `run.bat` (Windows).

Requires Java 9: to install Java SE 9, get the [Windows/macOS JRE installer](#) or run `apt-get install openjdk-9-jre` on Linux.

NEW The [Sirio library](#) is now available! The library implements the symbolic calculus and analysis methods of ORIS. STPN models can be exported from the GUI editor as "Java code" and analyzed in Sirio to conduct parametric studies.

Branch: **master** | New pull request

[Find File](#) | [Clone or download](#)

lauracarnevali	Update README.md	Latest commit overfree: 2 days ago
tool-paper-example	added se rejuvenation example	2 days ago
.gitignore	Initial commit	2 days ago
LICENSE.txt	Initial commit	2 days ago
README.md	Update README.md	2 days ago

README.md

A ready-to-use project on a software rejuvenation example

This repository provides a ready-to-use Maven project that you can easily import into an Eclipse workspace to experiment with the model of software rejuvenation discussed in the following paper:
"The ORIS Tool: Quantitative Evaluation of Non-Markovian Systems",
by Marco Paolieri, Marco Biagi, Laura Carnevali, Enrico Vicario,
IEEE Transaction on Software Engineering, to appear.

Just follow these steps:

1. **Install Java 9.** For Windows and macOS, you can download a [package from Oracle](#); on Debian unstable (sid) or testing (buster), or Ubuntu "bionic", you can just run `apt-get install openjdk-9-jdk`.
2. **Download Eclipse.** The [Eclipse IDE for Java Developers](#) package is sufficient.
3. **Clone this project.** Inside Eclipse:
 - Select `File > Import > Maven > Check out Maven Projects from SCM` and click `Next`.
 - If the `src URL` dropdown is grayed out, click on `src` Marketplace and install `src-e-git`. You will have to restart Eclipse (be patient...).
 - As `src URL` type: `https://github.com/oris-tool/tool-paper-example.git` and click `Next` and then `Finish`.

Your Eclipse project is ready! Just navigate to `src/main/java` and open `SoftwareRejuvenation.java` inside the package `org.oristool.examples`.