AvailSim4: Open-Source Framework For Availability and Reliability Simulations

Milosz Blaszkiewicz, Lukas Felsberger

CB Section Meeting, February 4th, 2025

Contents

Introduction

Methodology & Implementation

Resources

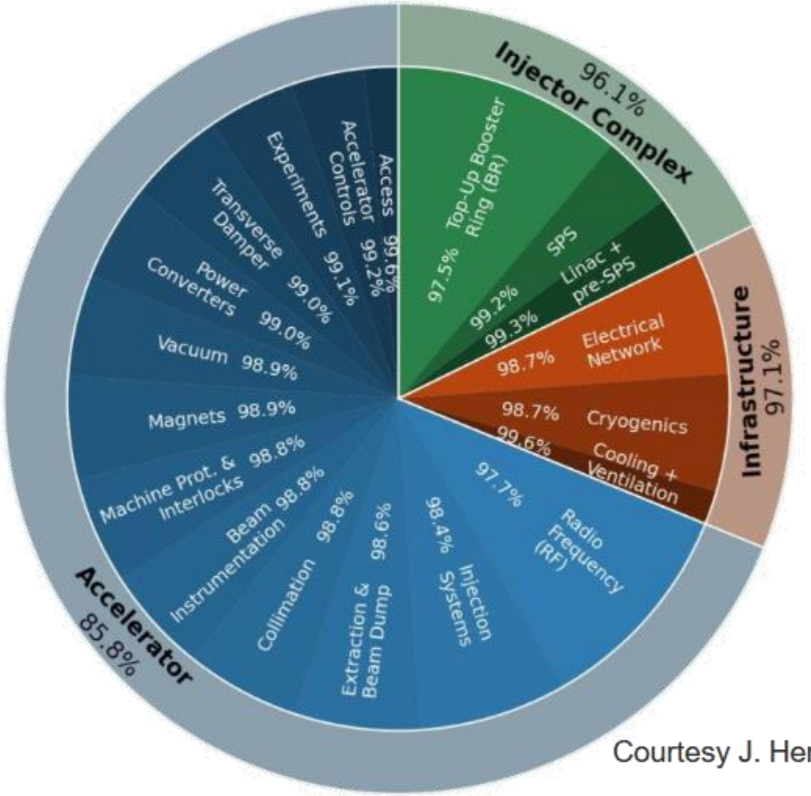
Conclusions

Contents

Introduction
RAMS simulations at CERN

- **Stochastic simulations for RAMS studies**
  - A broad range of tools is available to **estimate** or **predict** those metrics quantitatively.
  - Stochastic simulations:
    - **unparalleled flexibility,**
    - **straightforward translation** of a conceptual description into a model,
    - **highly realistic** representation of studied systems.

- **At CERN**
  - **Availability** concerns are relevant as the machine is an expensive project and downtime disrupts its scientific goals.
  - **Reliability** matters due to presence of systems that deal with large energy stored in the beams and magnets.



Courtesy J. Heron

Overview
What is AvailSim4?

Monte Carlo simulation framework for availability and reliability studies of complex systems.

Main characteristics of the framework:

- ✓ **Customizable models** for systems composed of **many sub-systems.**
- ✓ **Open source**; features tabular input & output, for **easy integration with other tools**.
- ✓ Parallelization capacity and distributed computing support for **large-scale simulations.**

Difficult to find in commercially availabile tools

AvailSim4

Project ID: 131878

Clone from **https://gitlab.cern.ch/availsim4/**
or

```
> pip install availsim4
```

The 4th take on the tool

Previous versions have been developed at other particle accelerator facilities.

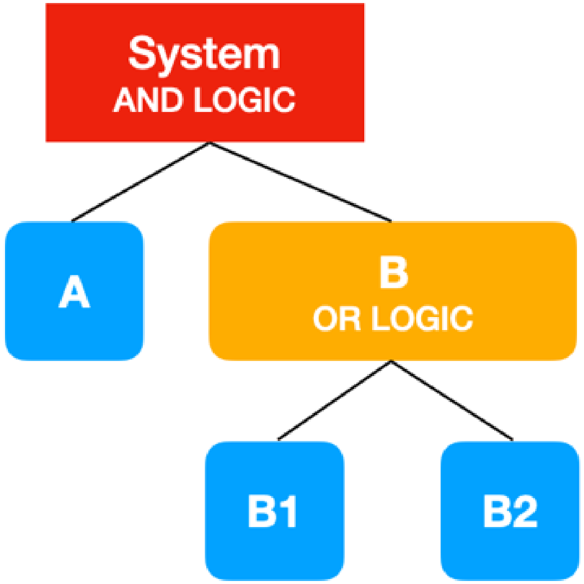
Current version:

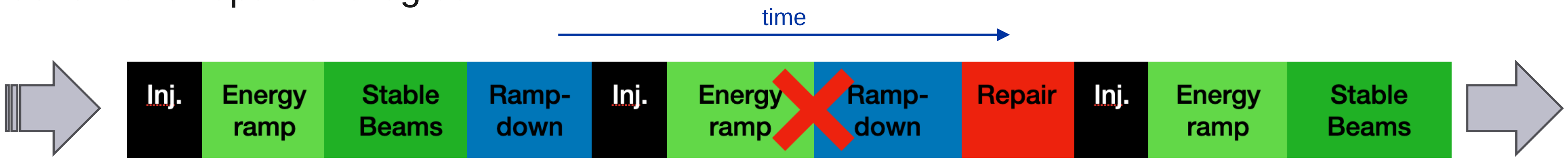
- written in **Python,**
- designed with **long-term maintainability** in mind,
- in use at CERN **since 2020.**

Contents

Algorithm & Implementation
Model Description

- **Models are made of components:**
  - **Basic** – elements with a failure mode
  - **Compound** – elements aggregating other basic and compound components into more complex structures
- **Component dependencies with logic operators:**
  - X out of Y, AND, OR…
- **Additional parameters:**
  - Phase-dependent failure and repair behaviour
  - Inspection and repair strategies





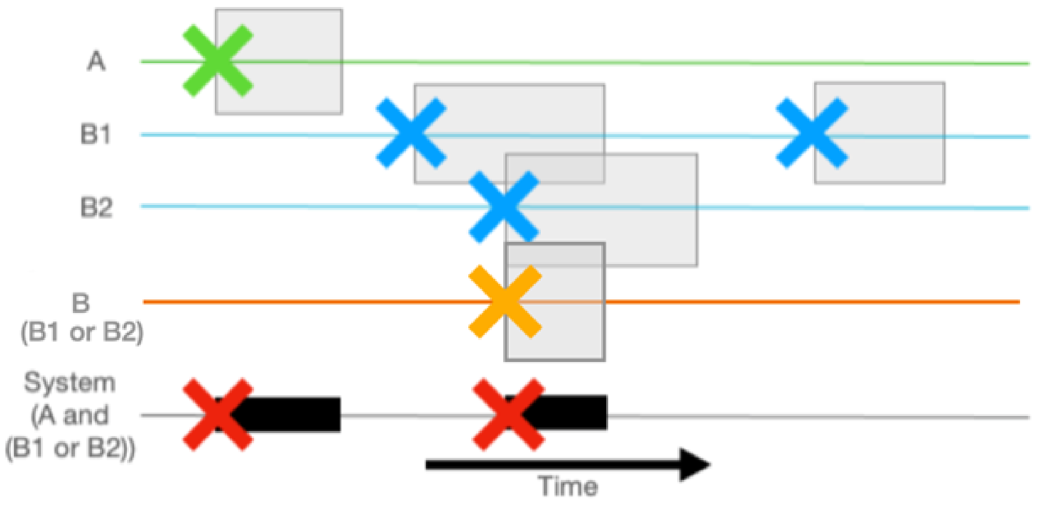
Algorithm & Implementation
Discrete Event Simulation & Monte Carlo

Discrete Event Simulation (DES):

- For driving the individual iterations.
- DES chosen to have maximum flexibility in modelling the system.

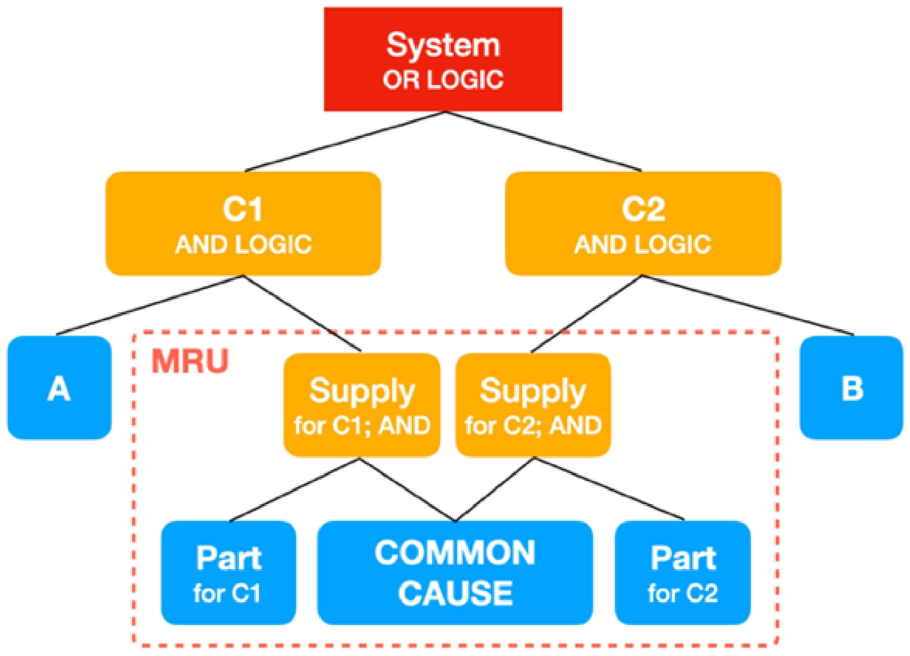
Monte Carlo (MC):

- DES performed repetitvely, each time sampling the desired probability distributions.
- The most flexible approach reflecting real-life events
  - Comes at the price of the slow convergence – for rare events, millions of iterations may be needed to obtain accurate results.



Selected built-in features
Support for complex models

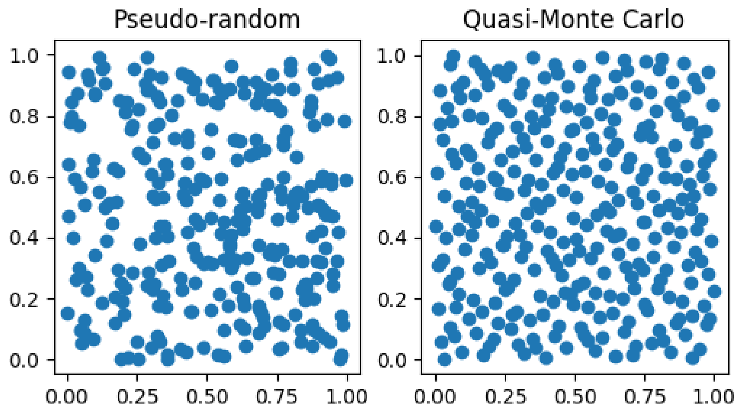
- **Minimal Replaceable/Swappable Units**
  - Failures of certain components may trigger repairs/replacement of others.
- **Shared children**
  - Parent components can be dependent on the same children.
- **Custom children logic**
  - Possibility to define custom advanced children logic through Python classes where other properties than a number of failures play a role.



Performance optimizations
Omptimized sampling

Disclaimer: not used much these days

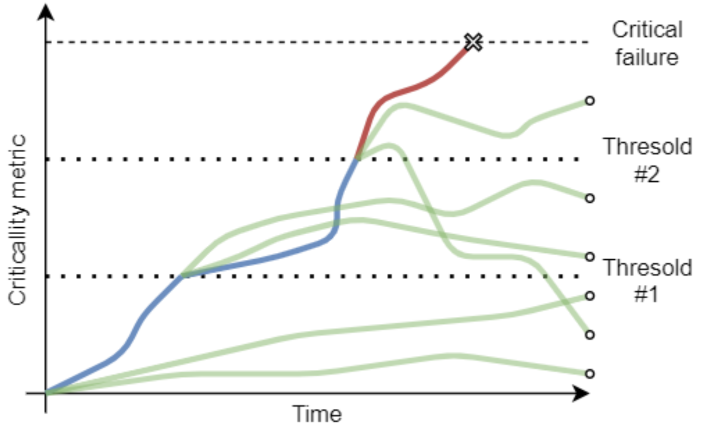
Quasi-Monte Carlo



- Improves convergence speed using **low-discrepancy sequences** (as opposed to pseudo-random in standard Monte Carlo).
- Reduces variance, increasing efficiency **"out-of-the-box"**.

Convergence of QMC closer to $O(1/N)$ instead of $O(1/N^2)$.

Importance Splitting
(RESTART and similar approaches)



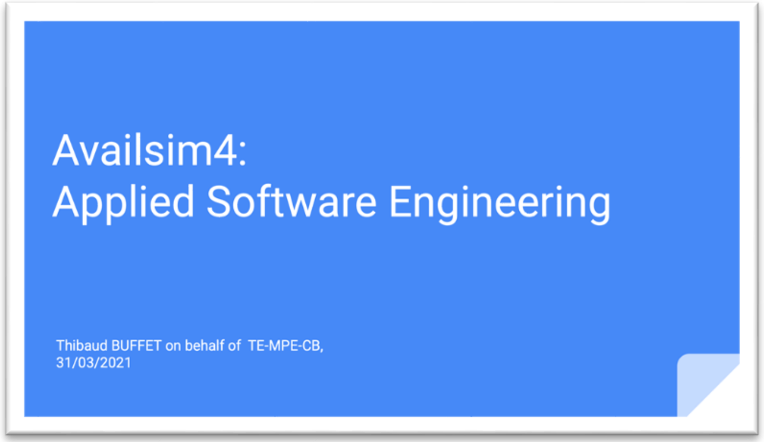
- Focuses on rare, critical events, through **splitting simulations** at crucial points.

Increased efficiency by **orders of magnitude** in synthetic test cases but **feasibility varies** per use-case and depends strongly on additional user input.

Code quality and long-term maintainability

Project features

- **High code quality:**
  - written in Python 3, with dependencies on well known libraries only
  - 11,000 lines, >200 tests in the Continuous Integration pipeline, 95% lines coverage
- **Getting started and contributing aided by**
  - User and developer guides.
  - Examples
- **Releases so far:**
  - 1st release in 2021, 2nd in 2023.
- **Available through PyPI and CERN Gitlab instance.**
  - Released under the GPL-3.0 licence.



See: TE-MPE TM Thibaud BUFFET 31/03/2021

Status	Pipeline	Created by	Stages
Passed 00:11:12 1 year ago	#142 fixing errors after rebase #4528659 142-cython 81b8c0be latest		
Passed 00:10:46 1 year ago	Merge branch '135-type_checking-jobs-re... #4516037 master e9048e3b		
Passed 00:11:06 1 year ago	#135 implementing reviewer's comments #4514612 135-type_checking-jobs-reporting-type-issues 724038c5		
Passed 00:16:40 1 year ago	#135 implementing reviewer's comments #4511893 135-type_checking-jobs-reporting-type-issues 9dbb76b4		

Contents

Introduction

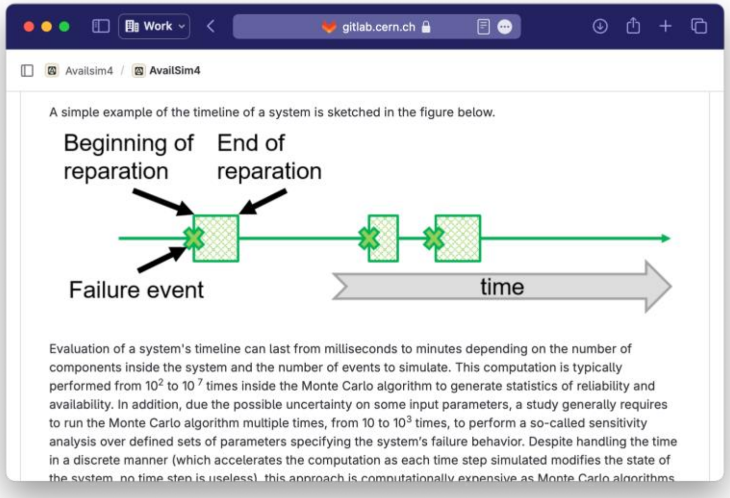
Methodology & Implementation

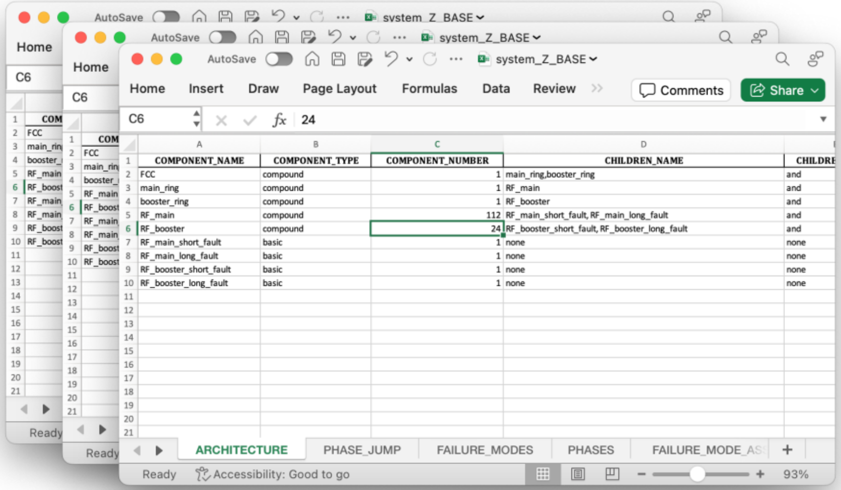
Resources

Conclusions

Getting started
Guides, examples, tools, etc.

- User guide: CERN Gitlab - AvailSim4 User Guide
- Examples: CERN Gitlab - Simple Examples for AvailSim4 beginners
- FCC study: CERN Gitlab - FCC-ee Sensitivity Availability Study
- Custom children logic showcase: CERN Gitlab - Dynamic Compensation Study
- HTCondor Post Processing: CERN Gitlab - Scripts to facilitate running on HTCondor





Release 2.2 (December 2024)
What's new?

New probability laws: shifted exponential and exponentiated Weibull (Gitlab - Issue 185)

Refactoring of the RCA code: doesn't create overhead when not used and stored in a memory-efficient way (Gitlab - Issue 184) and better phase treatment (Gitlab - Issue 184, 140)

HTCondor runner improvements (for using AvailSim4 as a module and from other scripts) (Gitlab - Issue 182)

Optional columns: all columns with default values no longer need to present (Gitlab - Issue 178)

Deployment: pyproject.toml instead of setup.py, fixed dependency requirements (Gitlab - Issue 181); AvailSim4 on CVMFS (Gitlab - Issue 166)

Documentation improvements and example notebooks (Gitlab - Issue 160, 181), typo and minor bug fixes (e.g., Gitlab - Issue 175)

Contents

AvailSim4 in the next years

Ensuring software remains functional and user-friendly

Ensuring performance in CERN environments (SWAN, HTCondor)

Maintaining GitLab CI pipelines

Providing support for availability studies

Conclusions

- **Advanced simulation tools are indispensable in detailed availability and reliability studies.**
- **AvailSim4 delivers an open-source solution, which:**
  - works with complex models,
  - is easily interfaced with other tools,
  - supports large-scale simulations via multi-core and multi-node computing.
- **Has been used in availability (LHC, FCC-ee, MYRRHA, etc.) as well as reliability (LHC Energy Extraction, LHC's Safe Machine Parameter system, etc.) projects.**
- **Explore at: gitlab.cern.ch/availsim4 (open access)**

@ CERN Gitlab

home.cern

Comparison with other tools
How does AvailSim4 compare to the others?

- **One of a few open-source projects in this area:**
  - Released under the GPL-3.0 licence.
- **Popular programming language (Python 3),**
  - Enables numerous extensions and adjustments in a straightforward manner,
  - Lowering the threshold for potential contributors.
  - Access to many other open-source libraries (such as QMCPy).
- **Ease of integration with other tools and automated pipelines through tabular input/output.**
- **Rare-event simulations: specialized techniques as well as focus on parallel and distributed computing.**
- **List of alternatives provided in the paper.**
- **Lack of GUI:** may be intimidating at first.

When to use AvailSim4?

Is modelling of **redundancy**, **demand**, **repair** required?

NO → **Analytical methods**

- Limited by **complexity of the equations**, at a certain point computer aid is necessary
- May quickly result in **sets of convoluted formulas** difficult to comprehend
- **Fast** to compute

YES → **Stochastic methods**

- **Very flexible**, potentially antyhing can be included in the simulation
- **Easy to understand** models
- Computationally **slow**