Advances in Reliability, Safety and Security, Part 4 – Kołowrocki, Dąbrowska (eds.)
© 2024 Polish Safety and Reliability Association, Gdynia, ISBN 978-83-68136-16-6 (printed), ISBN 978-83-68136-03-6 (electronic)

Advances in Reliability, Safety and Security
ESREL 2024
Monograph Book Series

AvailSim4: Open Source Framework For Availability And Reliability Simulations

Milosz Blaszkiewicz, Andrea Apollonio, Thibaud Buffet, Thomas Cartier-Michaud, Lukas Felsberger, Jack Heron, Jan Uythoven, Daniel Wollmann

CERN, Geneva, Switzerland

Abstract

Reliability, availability, maintainability, and safety (RAMS) of a system are key performance indicators. Availability simulations enable a nuanced understanding of the system's behavior under diverse conditions. This requires flexible, scalable, and performant simulation tools. AvailSim4 is a software framework for availability and reliability simulations of complex systems. Written in Python, it is a versatile, scalable, and user-friendly tool combining the Monte Carlo approach with the Discrete Event Simulation method. It enables detailed studies of customizable models for many-component systems, accounting for complex fault dependencies, inspection policies, adaptive model behavior according to operational phases and phase-transitions, and advanced repair policies. It is open source and features tabular input/output, allowing its integration with other tools. To handle large simulation models, AvailSim4 provides parallelization for computing grids as well as optimized sampling methods for increased computational efficiency. Application of the tool provides a comprehensive set of statistical insights allowing for the availability optimization of complex dynamic systems. It has demonstrated enormous value to reliability and availability assurance at CERN for particle accelerators and their sub-systems. This is highlighted in the example of the availability study of the CERN Future Circular Collider.

Keywords: reliability, availability, stochastic simulation, Monte Carlo, Discrete Event Simulation, simulation software, open-source

1. Introduction

Reliability and availability are key factors of system performance. If not properly considered, failures and downtime may lead to underperformance, financial losses and, eventually, premature end of life. However, if adequately treated, significant gains may be achieved using redundancies where required, optimal spare parts management, and appropriate intervention procedures.

Tools to evaluate reliability and availability can be categorized as *analytical* or *stochastic* models. Analytical models, such as Fault Tree Analysis (FTA) and Failure Modes, Effects and Criticality Analysis (FMECA), deconstruct failures and their interdependencies into trees and fault paths, which may be weighted according to probability and criticality. These provide valuable insight. However, they fall short when covering holistic scope and full-lifecycle dynamics of large complex systems (Kamat et al., 1975). It is possible to develop a general mathematical model for reliability purposes; those, however, are limited in terms of the complexity and may deviate from realistic scenarios on account of simplifications required to produce the model. They often feature implicit assumptions, which need careful consideration when interpreting the results.

Stochastic simulation methods such as Monte Carlo make use of random sampling to analyse the probabilistic nature of real-world processes. They allow for nuanced understanding of system behavior, accounting for changing operational conditions and indirect consequences of failures. They can significantly enhance accuracy of predictions and assist the development of more robust and resilient systems. However, they are computationally intensive, requiring significant resources for large-scale computations (Shonkwiler and Mendivil, 2009).

CERN operates a vast particle accelerator complex comprising machines with varying sizes and complexity. Among them there is the 27 km long Large Hadron Collider (LHC) (Brüning 2004), which is the largest and most complex particle accelerator ever built. The LHC itself is composed of many systems responsible for injecting, circulating, accelerating, and colliding particles. These include many advanced technologies in extreme operating conditions, e.g. superconducting magnets, ultra-high vacuum, radiation-tolerant electronics, and particle detectors. Reliability and availability assurance are essential to ensure the effective exploitation of the LHC. They are also necessary for the development of new accelerator components, as the performance of existing machines is improved, and components are consolidated. In addition, next generation machines like the Future Circular Collider (FCC) (Benedikt et al., 2018) are planned, which is CERN's preferred candidate for the next generation of energy-frontier particle accelerators, with a circumference of 91 km.

This paper presents the design methodology of AvailSim4, a probabilistic framework using the Monte Carlo (MC) approach and showcases its benefits in real-world use-cases. It is an open source, versatile, scalable, and user-friendly tool that has demonstrated its value for reliability and availability assurance at CERN. AvailSim4:

- enables detailed studies of customizable models for many-component systems, accounting for complex fault dependencies, inspections, operational phases, and replaceable structures.
- is open source and features tabular input/output, allowing integration with other tools.
- handles large simulation models, allowing parallelisation with computing grids as well as optimised sampling methods for increased computational efficiency.

The remainder of this paper is laid out as follows. Section 2 gives an overview of comparable simulation tools and highlights the potential shortcomings targeted by AvailSim4. Section 3 outlines the design requirements emerging from this comparison and relevant experience in availability studies at CERN. Section 4 outlines the algorithm, implementation, and most relevant features of AvailSim4. Section 5 shows the added value of AvailSim4 using the FCC-ee availability study as an example. Section 6 summarises overall content and conclusions.

2. Related work

The field of many-component availability and reliability simulations has seen multiple commercial and open-source software solutions, usually well-integrated within broader families of RAMS tools. Typically, these solutions complement analytical methods provided by these tools.

Formulation of the model is usually done within the software by means of graphical inputs of fault trees or reliability block diagrams, incorporating additional parameters of stochastic simulations – see (Penttinen et al., 2019; ReliaSoft, 2023; AvSim, 2023) for examples. In some cases, systems are represented with colored Petri nets (Realist, 2023; Robidoux et al., 2010). There, the emphasis is shifted from individual components towards state changes within the system.

A common denominator for the referenced reliability and availability simulation tools is the use of Monte Carlo algorithms. The inherent limitation of slow speed is addressed by limiting the scope of the simulations, runtime optimization, and the use of efficient computing languages.

Advantages and disadvantages of specific software solutions have been discussed in detail in (Bhattacharyya et al., 2012). Although there has been substantial progress in this field, certain challenges remain. A notable drawback of many existing tools is their proprietary nature. They are licensed, closed-code commercial software. Building extensions is difficult if not impossible in such cases, while interaction with the software is mostly restricted to manual inputs. Automation of workflows and integration of large volumes of results from external sources poses a significant challenge. Furthermore, outputs generated by these tools remain within the ecosystems of their parent software packages. At the same time, there have been few open-source software packages specifically targeting many-component reliability and availability simulations. An example is Stosim (Silkworth and Ormerod, 2018), which is open source and provides tabular input/output formats. However, its documentation is limited, and it is written in R, which is more difficult to maintain in a large software project.

The AvailSim tool aims to address shortcomings mentioned above. The software described in this paper is the fourth version in this family of frameworks. The first version of AvailSim was developed for SLAC/International Linear Collider (ILC) at Stanford University as a MATLAB program to simulate availability and effects of failures on performance of the accelerator (Sureda Pastor, 2013). It accepts a multitude of inputs, such as failures, maintenance, workforce needs. AvailSim2 was created for the International Fusion Materials Irradiation Facility (IFMIF) and added the support for additional factors such as operation parameters (Sureda Pastor, 2013). The next edition, AvailSim3, was developed for the European Spallation Source (ESS) project in collaboration with CERN (Bargalló et al., 2014). Developed in Python, it added a range of accelerator specific features, particularly the impact of a loss of system capacity on the performance of accelerator operation. Like this, it could, e.g., address the loss of accelerating cavities and their impact on the achievable particle energy.

AvailSim4 provides a complete simulation platform for availability and reliability needs, covering the whole range of functions and elements usually used in those fields. In addition, its input interface is open to both manual and programmatic access to allow unrestricted use of other tools or methodologies to communicate with the framework. This way, large simulation campaigns run on computing clusters or in cloud-based solutions are straightforward to achieve. On top of that, the framework may serve as a starting point for more tailored simulators by offering a high-quality, open, and object-oriented codebase, as well as extensive documentation, which makes it easy to interface with custom modules.

3. Requirements

Based on experience with previous versions of AvailSim and the above survey of existing tools, a set of requirements was derived for AvailSim4.

R1: The first and main requirement is to be able to perform user-defined analyses in the RAMS domain for any area of application. Its main functionality is to provide calculations of availability or reliability for the modelled system and its sub-systems. Models must be easy to define and adapt to real-world studies, with integrated support for relevant properties of simulated systems. This encompasses critical elements in RAMS analyses, such as failure modes, individual and group repair strategies, inspections, and operational phases simulating different loads or behaviours of the system. In addition, it should allow for advanced cases such as complex logic driving failure events. Although the primary target area of application is accelerator-related, the framework needs to be sufficiently abstract to handle any type of a technical system, given the variety of systems involved in accelerator technology.

R2: The second requirement is that the software should have means to easily interface with external sources, aside from accommodating user-defined inputs. It should allow users to integrate results of analyses performed elsewhere as well as data from various data sources, such as databases, fault tracking systems, etc.

R3: The third requirement relates to scalability and performance. The simulation engine must be designed for efficient handling of large-scale simulations and provide easy use of computing clusters. Advanced algorithms to optimize Monte Carlo, such as quasi-Monte Carlo, importance sampling or importance splitting, should be supported in the framework for rare-event simulations, as this is very relevant for reliability studies of safety-critical systems. The methods should be able to optimise the computational workload mostly automatically in simulations.

R4: The fourth requirement pertains to versatility, flexibility, and, in consequence, extensibility of the proposed software. The code should be open-source and developed with a wider user community in mind, acknowledging potential long-term benefits from internal and external contributors. This should be facilitated by a broad and generic application scope, as well as use of accessible, popular, and widely accepted programming languages and tools. Maintaining the tool as open-source project ensures that it can benefit from collaborative contributions, transparency, and the ability to adapt and customize the tool to specific research needs. Open-source projects foster knowledge sharing, enabling researchers to scrutinise, validate, and improve the algorithms and methodologies collaboratively. This allows for leveraging continuous refinements based on shared expertise.

R5: The fifth requirement pertains to accessibility and maintenance of the project, which requires comprehensive documentation. A user guide and other appropriate manuals should facilitate a swift and accurate introduction to the tool as well as maintain a high-quality record of available features and options. Prioritising a clear and concise user guide, well-documented code, and providing overviews of both simple and advanced examples is crucial, especially in high-turnover developers' and users' environments. Moreover, it is desirable for the project to be maintained in an accessible versioning system to ensure transparency, traceability, and efficient collaboration.

To address these requirements, AvailSim4 was written in Python utilizing object-oriented programming principles. It incorporates a model-agnostic and high-quality simulation codebase so that it can serve as a platform for a variety of availability and reliability simulations in the long-term. It provides a unified and cohesive approach to simulations, a crucial aspect in scientific and high-turnover user and developer environments. Being highly portable, it can run on local computers and computing grids alike. In addition, the software offers support for more advanced statistical methods, such as quasi-Monte Carlo sampling and importance splitting, to ease the computational burden of rare-event simulations. The following sections explain these aspects in more detail.

4. Methodology

4.1. Model Definition

The AvailSim4 framework represents systems as compound aggregations of components. An example of a simple system is shown in Figure 1a. *Basic components* (“A”, “B1”, “B2”) have an assigned failure mode, defining properties of their failure and repair probability distributions. The status of each *compound component* ("System" and "B") is defined by a logic depending on the fault status of its children. In Figure 1a, the compound System follows as "AND" logic for components "A" and "B". "B" follows an "OR" logic of components "B1" and "B2".

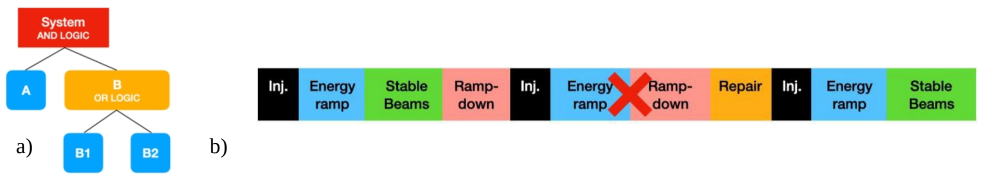


Fig. 1. (a) Example of a simple system with basic components A, B1, B2 and compound components System and B; (b) Succession of phases in AvailSim4 Monte Carlo iteration.

The mechanism defining the status of *compound components* is defined in terms of the number of subcomponents required to work. For instance, in a component with 3 children – 1 out of 3 is equivalent to having 3 redundant paths, 2 out of 3 stands for partial redundancy with one path acting as a back-up, 3 out of 3 means that there is no redundancy. AvailSim4 allows also to define a *custom children logic*. This feature is discussed in greater detail in Section 4.5.

During the simulation, the statuses of basic components are decided according to events generated from random samples. The random sampling is associated to stochastic failure and repair laws defined by the user for each element. Whenever a new event occurs, the status change of a component is propagated from the basic element upwards, updating statuses of all compound components. In the example of Figure 1a, the status of the *compound component* "System" depends on the statuses of its children: A and B, both required to work. A is a basic component, therefore its failure means that "System" fails too. Once it is repaired, the system operates again. The compound component B is made of two redundant basic components B1 and B2. Only when both fail, the compound "B" fails and with it "System" as well.

Operational phases are one of the fundamental parts of AvailSim4 simulations. They are essential in RAMS studies, as complex machines usually operate in different modes, resulting in different equipment stresses – modifying the probabilities of failure. The factor which complicates the analytical calculations is that phases may depend on the status of the system. E.g., a factory experiencing a stop will likely have to enter a restart procedure rather than simply continuing once the fault is repaired. In the simplified model of an accelerator such as the LHC, the following standard phases can be defined: injection, energy ramp, stable beams, ramp-down. Figure 1b shows the typical succession of phases in a collider like the LHC. Without failure the four phases occur one after the other guided by succession rules. If a failure occurs, (indicated by the red cross) the succession is disturbed. In the example a failure occurs during an energy ramp, the next phase will be a ramp-down, followed by a repair and the restart of the standard sequence of phases.

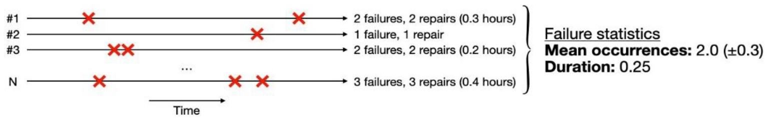

#1
#2
#3
...
N
2 failures, 2 repairs (0.3 hours)
1 failure, 1 repair
2 failures, 2 repairs (0.2 hours)
3 failures, 3 repairs (0.4 hours)
Time
Failure statistics
Mean occurrences: 2.0 (±0.3)
Duration: 0.25

Fig. 2. Illustration of N Monte Carlo iterations. Failure events are represented by crosses. Each iteration provides statistics of number of occurrences and duration of events (failure and repairs). The statistical summary provides insights for each type of events.

4.2. Algorithm

The Monte Carlo algorithm in the AvailSim4 framework operates by generating multiple *timelines*, each representing a separate and independent simulation of a *system lifetime*. Figure 2 presents a graphical representation of this concept. Randomization ensures a diverse set of potential outcomes. After conducting a comprehensive series of simulation runs, accurate estimations of mean, variance, and confidence intervals for the occurring events can be provided. The determination of sufficiency is a consideration of convergence, where additional simulations beyond a certain point yield negligible improvements in the precision of the estimates. The set of results typically describes events such as failures, repairs, inspections, and phase changes.

The Discrete Event Simulation method is used for executing *individual timelines*. It is implemented in the variant known as the three-phased approach (Pidd and Cassel, 1998).

The simulation terminates either when there are no more events to execute, or the next selected event is scheduled after the predefined simulated lifetime.

Figure 3 contains an example *timeline* of events for the simple system shown in Figure 1a, decomposed into timelines of events of individual components. Each iteration of the simulation starts with components in operating, non-failed state. A sample is drawn for each *event* which is valid at a given moment. Events are failures of components, repairs for failed components, etc. In the illustrated case, a failure of component “A” is occurring first. The simulation timer is updated to match the sample and the failure event is triggered. It adds a repair event with its length defined by another random number. Those samples are renewed each time a component changes its status. The failure of component “A” causes the entire “System” to fail, since “A” is necessary for it to operate. After it is repaired, the system is operating again. Then a failure of component “B1” occurs. It is one of the two redundant parts of component “B”, so it is degraded, but does not fail. When a failure of the component “B2” occurs before the other component is repaired, the whole component “B” is failed, causing the second failure of the “System”.

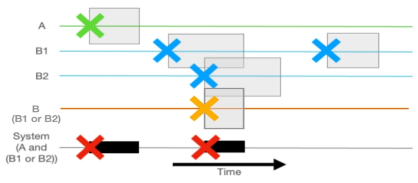
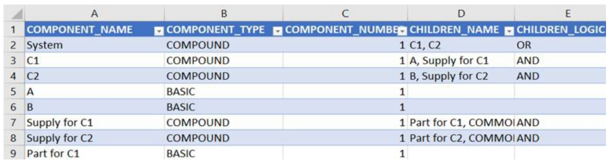


Fig. 3. Timelines of events comprising an individual system lifetime.

4.3. Implementation

This section outlines in a concise way the chosen code implementation strategies for AvailSim4. The input and output format, the coding structure, the development process, and key considerations that underpin the overall functionality and quality are discussed.

AvailSim4 is a command-line tool, which uses input files prepared in external spreadsheet applications. This has been done for simplicity, as system experts providing input data can easily enter it in this format. Moreover, tabular data can be generated and modified in a programmatic manner, which is important for automation. The template input files are well defined and follow the structure which is shared with the code and documented in the user guide. Figure 4a shows an example of a configuration of a system. A block diagram of this system is shown in Figure 4b. It features five basic components, grouped into two high level compound components: C1 and C2. Each compound component has its own separate basic component (A, B) and a joint component “power supply”. The “power supply” has a common part (“common supply”) and parts dedicated to specific sub-systems (A_SUPPLY, B_SUPPLY).



	A	B	C	D	E
1	COMPONENT_NAME	COMPONENT_TYPE	COMPONENT_NUMBE	CHILDREN_NAME	CHILDREN_LOGIC
2	System	COMPOUND	1	C1, C2	OR
3	C1	COMPOUND	1	A, Supply for C1	AND
4	C2	COMPOUND	1	B, Supply for C2	AND
5	A	BASIC	1		
6	B	BASIC	1		
7	Supply for C1	COMPOUND	1	Part for C1, COMMO	AND
8	Supply for C2	COMPOUND	1	Part for C2, COMMO	AND
9	Part for C1	BASIC	1		

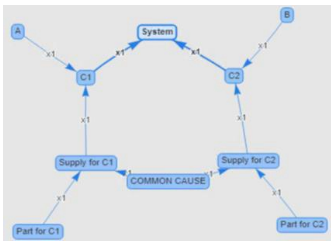


Fig. 4. (a) Example contents of the Architecture sheet of the AvailSim4's input file;
(b) real-time visualization add-in showing the block diagram as a graph of components.

The choice of Python as the programming language was driven by its extensive use in the scientific community. This choice reflects the general trend within scientific computing (Millman and Aivazis, 2011), as it offers integration with many open-source libraries, facilitating transparency, and increasing the potential for external contributions. With the advantages of simplicity and expressiveness of the language, comes the cost of reduced computing efficiency. Python, an interpreted language, performs worse than most compiled languages, although an increasing number of optimisations is addressing this issue (Nagpal and Gabrani, 2019). This aspect is discussed for AvaiSim4 in Section 4.4.

High quality of the developed code is realized on various levels: the development is subjected to a thorough code review process performed by software developers external to the team and a comprehensive set of 141 automated checks running in a continuous integration pipeline. The code review ensures the adherence to coding standards, fostering collaboration among developers, and strongly promoting the overall quality. The continuous integration pipeline comprises unit, end-to-end and integration tests as well as tools measuring code quality metrics. Those programmatic tests are run after every change to the code, determining if all functionalities remain intact. The tests, aside from preventing unwanted changes, help achieving backwards-compatibility, further ensuring that early models can work with newer releases of the framework. The code quality tools keep track of metrics such as testing coverage, conformity with coding standards such as PEP8 and code anomalies.

Ten developers have been involved in the creation and maintenance of the code over the last four years. It has been used in at least four well documented cases. Aside from the later mentioned FCC project, it was also the main tool for HL-LHC Energy Extraction study (Blaszkiewicz et al., 2022), Solid-State RF amplifiers (Felsberger et al., 2021), availability study of MYRRHA accelerator driven system (Felsberger et al., 2023, 2024). It is also actively used for other ongoing reliability and availability studies at CERN.

Thanks to the early adoption of the open-source strategy – AvailSim4 does not depend on internal CERN software packages nor proprietary solutions. As such, AvailSim4 can be easily integrated into any environment and avoids "vendor lock-in" – promoting free access and interoperability between various ecosystems. Moreover, it makes the project more sustainable in the long-term, further reinforcing its longevity.

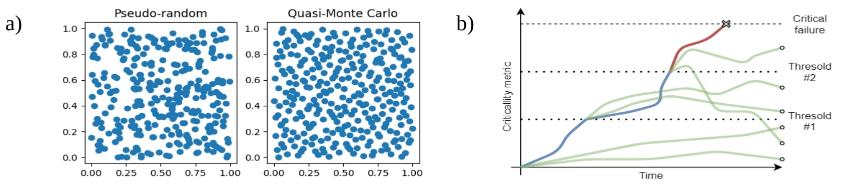


Fig. 5. (a) Plot of two-dimensional quasi-Monte Carlo samples compared with standard pseudo-random number generator; (b) an illustration of example evolution of restarts in the Importance Splitting algorithm

4.4. Optimized sampling approaches

A frequently raised challenge associated with Monte Carlo methods is their propensity for being computationally intensive. This becomes particularly pronounced in the context of reliability simulations for safety-critical systems, in which events of interest occur very rarely. Using the standard Monte Carlo algorithm to observe such events, the overall execution time may surpass reasonable resource and time constraints defined for a study if executed on personal computers.

To address this challenge, the AvailSim4 framework is designed to function within *High-Performance Computing (HPC) grids*. Monte Carlo simulations are easily parallelisable since iterations are entirely independent from each other. Consequently, simulations can be efficiently executed in batches across separate computing nodes. The final step, involving the processing of results from individual instances, is deferred until all instances have finished their simulations.

For users at CERN, AvailSim4 has a built-in support for the CERN grid (Thain et al., 2005). It encompasses both assistance at submitting the computation jobs to the grid and processing the results from individual instances. Using this feature, users can runAvailSim4 simulations simultaneously on hundreds of computing nodes.

Aside from using more computational power, the AvailSim4 framework also has implemented two advanced sampling methods for speeding-up the simulations by increasing computational efficiency: *quasi-Monte Carlo sampling* (Owen, 2003) and *Importance Splitting* (Garvels, 2000).

The first method is based on replacing a sampling provided by a pseudo-random number generator with one that chooses the samples more efficiently. As illustrated in 5a, the pseudo-random number generation is generally not as efficient at uniform coverage of the problem space as it could be. In a plot of two-dimensional samples in a plane, one can see clustering in some places and empty irregular areas in others. *Quasi-Monte Carlo* counters this by producing samples based on low-discrepancy sequences. Although not random by default, they are a better choice as sources of random numbers, leading to decreased variance of results. The problem space can be covered more evenly with less samples. AvailSim4 utilizes an open-source library (Choi et al., 2021) allowing users choose the Quasi-Monte Carlo sampling for improved sample efficiency (Blaszkiewicz, 2022).

The second method, *Importance Splitting*, is concerned with the simulations themselves. Since not all simulations will eventually produce relevant events, it is more efficient to guide simulations towards relevant events and discount their probability of occurrence by factoring in that relevant events were oversampled. This means that each iteration going through a system lifetime must keep track of a certain case-specific "criticality" metric. When it reaches a pre-defined threshold, the iteration is stopped and repeated in multiple copies going onwards in the simulation time. Each iteration can have multiple thresholds, meaning that the branching will occur many times. Promoting "promising" iterations will lead to increased likelihood of observing the event of interest and estimating its probability. Conversely, this also means that rare scenarios may be omitted if the criticality metric is defined incorrectly.

Figure 5b shows an example of how criticality metric evolves in simulations started in only three instances in the first level. The criticality metric can be, e.g., a number of faults in the system. When in one iteration the criticality metric reaches the first threshold, that simulation is stopped and restarted in three copies from that point onwards. One of them reached another threshold and is repeated three times again.

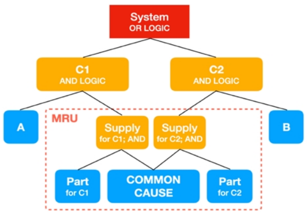


Fig. 6. Example of a model utilizing shared children feature.

4.5. Selected features of AvailSim4

The component structure in AvailSim4 supports the *shared children* feature. This means that each component may appear as a child of more than one parent. In Figure 6 this is the case for the "Common Cause" component, which is shared by the "supply for C1" and "supply for C2" components. With this feature, the structure composed in AvailSim4 is no longer a tree but a directed acyclic graph. The root of that graph represents the entire system. Connections between components can create an arbitrarily large structure of dependencies.

A common characteristic of repairable systems is that groups of components, a tray, or a box, are swapped in case of a failure instead of repairing individual components in-situ. This can be modelled by *minimal replaceable units* (MRU). In Figure 6, the MRU is defined for the power supply element used by components C1 and C2. The description of the system in the input file is presented in Figure 4a. The system is built of two compound redundant parts: "C1" and "C2". Each has a specialized basic component (respectively "A" and "B") but also depends on a "Supply", which has a shared part "Common Cause" and parts specific for "C1" and "C2". Whenever any of those fails, they will be replaced together, effectively removing possibly hidden failures in the MRU.

Periodic inspections are frequently featured in RAMS analyses. In redundant systems, failures may not always be visible, and their presence is revealed only when a suitable inspection is performed. Inspections are one of the key maintenance elements preventing critical errors of complex systems. They ensure that redundancy can be maintained, drastically decreasing the likelihood of an overall critical damage. In AvailSim4, each failure mode can have an associated inspection interval, at which it can be discovered and resolved.

Custom children logic is another advanced feature included in the simulation engine which enables modelling fault dependencies in relation to other operational conditions (such as phases). User may use it to define an advanced logic deciding on a compound component's failure by expressing it in a Python class implementing a defined structure. This way, models may express redundancies where only certain configurations of faults among sub-systems can be tolerated or change depending on a specific phase. An example of this is dynamic compensation of chains of RF systems in particle accelerators (Felsberger et al., 2023) and (Felsberger et al., 2024).

Root cause analysis is a tool to gather additional information about events in the simulations. The feature takes a snapshot of all components' state in the model when user-defined circumstances arise. For instance, if the component C2 in Figure 6 fails, it might be relevant to understand whether it is more often caused by the power supply dedicated to the C2 component or the component B. In this case, the root cause analysis allows users to save states of all components in the system when C2 fails. This feature is also very useful for the debugging of models. As an addition, AvailSim4 project features also an add-in to Microsoft Excel for supporting model creation. It features input structure creation, automatic suggestions for references to elements of other tables, a compact user guide accompanying specific worksheets and a real-time visualization of the component graph. A screenshot of the extension is presented in Figure 4b.

5. Use-Case: The Future Circular Collider (FCC) RF Systems

This section provides a recent example to showcase an application of AvailSim4 to RAMS analyses. The FCC is CERN's leading proposal for the next generation of energy-frontier particle accelerators. The AvailSim4 framework was used to study certain availability aspects as part of a feasibility study.

The FCC is proposed to have a circumference of 91 km, adjacent to the 27 km LHC, which would make it the largest particle collider ever built. Each year, 185 days are scheduled to physics, of which 80% must be spent at nominal parameters to achieve physics goals (Benedikt et al., 2019). The LHC was available for 77% of the period of 2016-2018 (Todd et al., 2019). Considering additional challenges in maintaining the *FCC*, like its size, complexity, and ambitious technical challenges, availability is a significant risk to its objectives.

FCC operation is planned in two stages, the first being an electron-positron collider (*FCC-ee*) starting around 2040. It will have four energy modes in its projected 15-year lifetime, labelled Z, W, H and $t\bar{t}$, which will differ in operational requirements. The *FCC-ee* design can be divided into two functional systems: the *injector complex*, which produces and accelerates the particles, and the *main colliding ring*. Assuming no external delays, planned operation consists of six phases shown in Figure 7:

1. *Set Up:* Magnets are cycled, RF systems are set and equipment is prepared for injection of particles, 10 minutes.
2. *Fill:* Particles are injected into the main colliding ring, 5 minutes.
3. *Adjust:* Final adjustments are made for collisions, 10 minutes.
4. *Physics + Top-Up*: Particle collisions begin, producing luminosity. Particles that are "burned off" in collisions are replenished by continuous new beam injections in the "top-up" procedure. Hence, the accelerator can remain in this phase until it is deliberately ended by the operations team or by an equipment failure.
5. *Burn Off:* If the injector complex fails during phase 4, the main colliding beams can be maintained with decaying luminosity for approximately 60 minutes, after which the accelerator goes down for repair.
6. *Down for Repair:* If the main colliding ring fails in any phase or the injector fails during phases 1-3, the accelerator is stopped for repair.

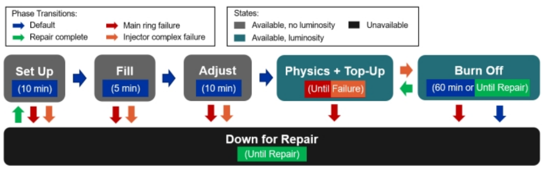


Fig. 7. Schematic representation of phases and their succesion in the model of FCC RF.

In Figure 7, the standard succession of phases is shown by the blue arrows. Deviations due to faults and failures are indicated by orange (injector complex) and red (main colliding ring) arrows and they may occur in any phase except *Down for Repair*. Following a failure, a repair process restores the default phase sequence according to the green arrows. The whole sequence is implemented in AvailSim4 using its Phase features.

Table 1. Energy modes, applicable redundancies with corresponding numbers of cavities; from (Raubenheimer, 2023).

Energy Mode	Z		W		H		$t\bar{t}$	
Beam Energy [GeV]	45.6		80		120		182.5	
Redundancy	none		none		10%		10%	
	Main ring	Injector complex	Main ring	Injector complex	Main ring	Injector complex	Main ring	Injector complex
# Cavities	112	24	264	56	264	112	752	600

The injector complex and main colliding ring are composed of many constituent systems, and availability must be assessed individually for each one. The Radio Frequency (RF) was chosen as the first. It is responsible for accelerating particles to the nominal collision energy and topping up the energy emitted by the large amounts of synchrotron radiation. The RF system has an availability requirement of 97.7% (Heron et al., 2023). It is composed of discrete cavity circuits in growing numbers for each energy mode, shown in Table 1, (Raubenheimer, 2023). Depending on the energy mode, there are between 136 and 1352 elements in the AvailSim4 simulation.

Theoretically, nominal energy can be preserved if no more than 10% of cavities are unavailable. Practically, this is feasible only for the H and $t\bar{t}$ modes, where the beam current is relatively low. This is represented in the AvailSim4 model by assigning children logic which allow for a ***degraded*** state with incomplete redundancy. In Z and W, a tripped cavity immediately dumps the beam to prevent beam induced damage to the cavity.

RF faults may occur in two types. *Short Faults* are reparable via remote reset. In $H, t\bar{t}$ modes, they can be repaired without changing phase provided the total number of tripped cavities does not exceed 10%. *Long Faults* require human intervention and cannot be repaired while the beam is running but only when the accelerator is Down for Repair. This is modelled by combining children logic and phase features in AvailSim4.

Key simulation inputs are the four probability distributions governing the simulation models shown in Table 2. These are based on experience with comparable systems in the LHC (Heron et al., 2023).

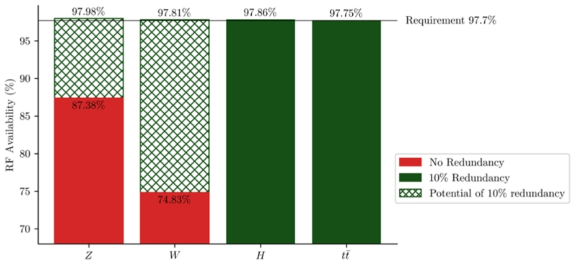


Fig. 8 Results of the FCC RF model simulations.

The simulation is a projection of the *FCC-ee* if reliability and repair time per RF circuit stay as they are in the LHC. Each energy mode was simulated through 100 years of operation, with results shown in Figure 8. The drop in availability from Z to W is explained by the greater number of cavities and therefore greater exposure to RF faults. While the number of cavities also increases through $H, t\bar{t}$, the 10% voltage redundancy in these modes keeps availability above the 97.7% requirement.

Projections for the Z, W modes show that RF cavities will lead to inadequate availability if no redundancy is present. However, cavities at all energy modes have the same 10% voltage margin. Therefore, the same redundancy seen in $H, t\bar{t}$ could in principle be used in Z, W modes. A candidate technology to achieve this is ferroelectric fast reactive tuning with an appropriate response time of $4\mu s$ (Kanareykin et al., 2023). The potential availability gain using this technology was simulated using the same AvailSim4 model, illustrated by hashed bars in Figure 8. A concerted R&D effort is now underway to demonstrate the technical feasibility of this solution.

Table 2. Probability distributions used in the model; unit: hours (h), where applicable.

Fault	Fault Dist.	Parameters	Repair Dist.	Parameters
Short Fault	Exponential	$1 - e^{-\lambda t}, \lambda \cong 1/2{,}476$ h	Exponential	$1 - e^{-\lambda t}, \lambda \cong 1/0.22$ h
Long Fault	Exponential	$1 - e^{-\lambda t}, \lambda \cong 1/3{,}594$ h	3-p. Weibull	$1 - e^{-\left(\frac{[t-\gamma]}{\lambda}\right)^k}, \lambda \cong 1.94\, h, k \cong 1.35, \gamma \cong 1.75$ h

6. Summary

AvailSim4 is an open-source tool offering a Monte Carlo-based simulation environment for availability and reliability studies. Its features were created with safety-critical systems for the accelerator community in mind, but the developed approach is general and can be directly applied to other application areas. As a successor to previous versions of AvailSim, it builds on top of extensive practical knowledge and addresses gaps identified in comparable tools. It is an open-source project developed at CERN and released under the GPL-3.0 license.

The framework allows for a significant degree of flexibility and integration with other tools. It facilitates in-depth examinations of tailored models for systems with numerous components, considering intricate fault interconnections, inspection strategies, adaptive model responses based on operational phases and transitions, and sophisticated repair policies. Tabular input format makes it easy to programmatically interface through custom user scripts with a wide variety of other software packages. Computational performance, aside from using state-of-the-art external libraries and optimisation techniques, is addressed by Quasi-Monte Carlo and

Importance Splitting methods to decrease high computational workloads for reliability simulations of safety-critical simulations.

To ensure the longevity of the project, extensive documentation and high-quality code have been applied from the very start of the project thorough code reviews and automated checks. The development concentrated in the open repository will continue to focus on addressing ongoing needs of the users.

AvailSim4 has been in use since 2019. It has demonstrated its ability to deal with real-world scenarios and studies, both at CERN and in other research institutions. It has been used in several well-documented cases of availability and reliability studies and it continues to be in use for future projects like the Future Circular Collider (FCC). As shown in this use-case, an AvailSim4 simulation can be instrumental to identify availability requirements of sub-systems for highly configurable operational scenarios.

References

AvSim, accessed 15.12.2023, https://www.isograph.com/software/availability-workbench/availability-simulation/avsim/, Isograph Inc.

Bargalló, E., Sureda Pastor, P.J., Manuel Arroyo, J., Abal, J. and De Blas, A., Dies, J., Tapia, C., Mollá, J. Ibarra, A. 2014. Availability simulation software adaptation to the IFMIF accelerator facility RAMI analyses. Fusion Engineering and Design 89, 2425-2429.

Brüning, O. (Ed.) et al. 2004. LHC Design Report Vol. 1 The LHC Main Ring.CERN Yellow Reports: Monographs. CERN, Geneva.

Benedikt, M. (Ed.) et al. 2019. FCC-ee: The Lepton Collider: Future Circular Collider Conceptual Design Report Volume 2. European Physical Journal: Special Topics 228(2), 261–623, 6 2019.

Bhattacharyya, S., Yedavalli, R.K., Kerby, J. and Mukherjee, A., 2012. Reliability Modeling Method for Proton Accelerator. Proc. IPAC'12, Louisiana, USA.

Blaszkiewicz, M. 2022. Methods to optimize rare-event Monte Carlo reliability simulations for Large Hadron Collider Protection Systems. University of Amsterdam, Netherlands

Blaszkiewicz, M., Apollonio, A., Cartier-Michaud, T., Panev, B., Pojer, M., Wollmann, D. 2022. Reliability Analysis of the HL-LHC Energy Extraction Systems. Proc. IPAC'22, Bangkok, Thailand.

Choi, S.-C. T., Hickernell, F. J., and Jagadeeswaran R. and McCourt, M. J., and Sorokin A. G. 2021. Quasi-Monte Carlo Software. arXiv.

Felsberger, L., Dorda, U., van de Walle, J., Uythoben, J., Wollmann, D. 2024. Simulation-Based Availability Optimization of Dynamic Fault Compensation for Particle Accelerator RF-Systems Applied to the MYRRHA Accelerator Driven System. Submitted at ESREL 2024.

Felsberger, L. et al., 2023. Quantitative availability modelling for the MYRRHA accelerator driven system. Proc. IPAC'23, Venice, Italy, 5114-5117.

Felsberger, L., Apollonio, A., Cartier-Michaud, T., Montesinos, E., Olivera, J. C., Uythoven, J. 2021. Availability Modeling of the Solid-State Power Amplifiers for the CERN SPS RF Upgrade. Proc IPAC'21, Campinas, SP, Brazil.

Garvels, M. J. J. 2000. The splitting method in rare event simulation. University of Twente, Netherlands

Heron, J., Felsberger, L., Wollmann, D., Uythoven, J., Rodriguez Mateos, F. 2023. Availability Targets Scaled According to Assurance Complexity in the FCC-ee. ATS Note: CERN-ACC-NOTE-2023-0020. CERN. Geneva. ATS Department.

Kamat, S. J., Riley, M. W. 1975. Determination of Reliability Using Event-Based Monte Carlo Simulation. IEEE Transactions on Reliability R-24(1), 73-75, April 1975.

Kanareykin, A., Ben-Zvi, I., Castilla, A., Freemire, B., Jing, C., Macpherson, A., Poddar, S., Shipman N. 2023. Ferroelectric fast reactive tuner for srf cavities - material properties and its applications. Proceedings of FCC Week 2023, London, United Kingdom.

Millman, K. J., and Aivazis, M. 2011.Python for Scientists and Engineers. Computing in Science & Engineering 13(2), 9-12

Nagpal, A., Gabrani, G. 2019. Python for Data Analytics, Scientific and Technical Applications. 2019 Amity International Conference on Artificial Intelligence (AICAI), Dubai, United Arab Emirates, 140-145.

Owen, A. B. 2003. Quasi-Monte Carlo Sampling. Jensen, H. W. (Ed.), Monte Carlo Ray Tracing: Siggraph 2003 Course 44, 69-88.

Penttinen, J.-P., Niemi, A., Gutleber, J., Koskinen, K. T., Coatanéa, E., Laitinen, J. 2019. An open modelling approach for availability and reliability of systems. Reliability Engineering & System Safety 183, 387-399.

Pidd, M. Cassel, R.A., 1998, December. Three phase simulation in Java. In 1998 Winter Simulation Conference. Proceedings 1, 367-371, Cat. No. 98CH36274. IEEE.

Raubenheimer, T. 2023. FCC Accelerator Overview. Proceedings of FCC Week 2023, London, United Kingdom.

REALIST, accessed 15.12.2023, https://www.ima.uni-stuttgart.de/forschung/zuverlaessigkeitstechnik/realist/, University of Stuttgart.

ReliaSoft, accessed 15.12.2023, https://www.hbkworld.com/en/products/software/analysis-simulation/reliability, Hottinger Brüel & Kjær.

Robidoux, R., Xu, H., Xing L., and Zhou, M. 2010. Automated Modeling of Dynamic Reliability Block Diagrams Using Colored Petri Nets. IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans 40(2), 337-351.

Shonkwiler, R.W., Mendivil, F. 2009. Introduction to Monte Carlo Methods. Explorations in Monte Carlo Methods. Undergraduate Texts in Mathematics. Springer, New York, NY.

Silkworth, D., Ormerod, J., 2015. Stosim, OpenReliability.org.

Sureda Pastor, P.J., 2013. Adaptation of the Availsim software to the IFMIF RAMI requirements. UPC, Escola Tècnica Superior d'Enginyeria Industrial de Barcelona, Departament de Física i Enginyeria Nuclear. Available at: http://hdl.handle.net/2099.1/20970.

Thain, D., Tannenbaum, T., Livny, M. 2005. Distributed Computing in Practice: The Condor Experience. Concurrency and Computation: Practice and Experience 17(2-4), 323-356.

Todd, B., Apollonio, A., Niemi, A., Ponce, L., Roderick, C., Walsh, D. 2019. LHC and Injector Availability: Run 2. 9th LHC Operations Evian Workshop, Evian Les Bains, France, 30 Jan – 1 Feb 2019, 35-50.